A Comprehensive Literature Review with Self-Reflection

Literature Review

October 16, 2025

Abstract

This literature review provides a comprehensive analysis of recent research in the field. The review synthesizes findings from 297 research papers, identifying key themes, methodological approaches, and future research directions.

Contents

1 Introduction

1.1 The Evolving Landscape of Software Development and AI

The advent of Artificial Intelligence (AI), particularly the rapid advancements in machine learning and generative models, is instigating a profound paradigm shift in software development. This transformation moves beyond incremental improvements, fundamentally reshaping traditional manual processes towards increasingly AI-assisted, and even autonomous, software creation ??. This evolution necessitates a critical re-evaluation of existing engineering practices, governance models, and the very tools and methodologies developers employ, highlighting AI's dual role as both a powerful enhancer of efficiency and a complex product requiring careful lifecycle management ??.

While AI and machine learning techniques have long been applied to specific software engineering tasks, such as defect prediction, effort estimation, and requirements analysis ?, the current wave, largely driven by Large Language Models (LLMs), represents an unprecedented integration across the entire Software Development Lifecycle (SDLC) ?. This pervasive integration is transforming how software is conceived, designed, coded, tested, deployed, and maintained. Developers are increasingly interacting with AI not merely as a tool, but as a collaborative partner, leading to new human-AI interaction paradigms that redefine productivity and creativity ?.

At a high level, the impact of AI spans several key areas:

- **AI-Assisted Development**: AI is augmenting human capabilities in various stages, from generating code snippets and completing functions to suggesting design patterns and refactoring code. This assistance aims to accelerate task completion and reduce cognitive load, leading to significant productivity gains ?. The emergence of "AI pair programmers" marks a significant shift in developer workflows, fostering new models of human-AI collaboration that will be explored in detail in Subsection 2.2.

- **Automated Software Engineering**: Beyond assistance, AI is enabling higher degrees of automation. This includes autonomous bug detection and fixing, automated test case generation, and even the orchestration of entire development processes through intelligent agents ?. These advancements pave the way for more efficient and reliable software systems, with specific applications in testing and quality assurance to be discussed in Section 4 and 5.

- **Requirements and Design**: AI is also beginning to influence the upstream phases of the SDLC, assisting in the interpretation of complex natural language requirements, generating design specifications, and ensuring compliance-by-design from the outset ?. This proactive integration of AI aims to embed quality and compliance earlier in the development process.

However, this transformative potential is accompanied by a new set of challenges and complexities. The development and deployment of AI-enabled systems introduce unique considerations that traditional software engineering practices may not adequately address ?. These include managing vast and often evolving datasets, ensuring the explainability and fairness of AI models, and addressing the inherent risks of bias and "hallucinations" in generative AI ??. The integration of AI also necessitates a re-evaluation of security practices, as AI-generated code can introduce vulnerabilities, and developers may exhibit overconfidence in AI suggestions, leading to less secure outcomes ??. These critical human factors and security implications will be further elaborated in Subsections 4.2 and 6.3.

The need for a more structured and disciplined approach to developing AI-enabled systems has become paramount ?. Organizations are exploring frameworks, such as TOGAF, to integrate AI effectively and manage the associated risks and challenges ?. This underscores the necessity for robust governance models and updated engineering practices that account for the unique characteristics of AI components, treating them not just as tools, but as integral, complex, and potentially opaque parts of the software product itself ??.

In conclusion, the evolving landscape of software development with AI is characterized by a dynamic interplay between unprecedented efficiency gains and the imperative to manage novel complexities. This section has introduced the broad strokes of this transformation, setting the stage for a deeper exploration of foundational AI capabilities in software engineering (Section 2), the critical need for trustworthy and responsible AI development (Section 3), and the specific applications and challenges of AI in ensuring software development compliance (Sections 4, 5, and 6). The subsequent sections will delve into the empirical evidence, theoretical frameworks, and practical implications of this ongoing revolution.

1.2 Defining AI for Software Development Compliance

The burgeoning field of Artificial Intelligence (AI) for software development compliance represents a critical and multifaceted domain, which this literature review approaches through a dual lens. At its core, "AI for Software Development Compliance" encapsulates two interconnected yet distinct dimensions: first, the strategic application of AI technologies to enhance and automate compliance-related tasks within the software development lifecycle (SDLC); and second, the imperative to ensure that AI systems themselves are developed and deployed in a compliant, ethical, and trustworthy manner. This dual perspective is foundational to understanding the landscape of research, challenges, and opportunities in this rapidly evolving area.

The first dimension, leveraging AI *for* compliance, focuses on the deployment of advanced AI technologies—including machine learning (ML), natural language processing (NLP), and generative AI (GenAI)—to automate, streamline, and improve the efficiency and accuracy of compliance activities throughout the SDLC. This encompasses a broad spectrum of tasks, such as automated security vulnerability detection in codebases **?**, ensuring adherence to complex regulatory mandates like data privacy laws **?**, and enhancing rigorous quality assurance (QA) processes. For instance, AI can analyze vast amounts of code, requirements documents, and design specifications to identify deviations from established security policies, predict potential defects, or even generate test cases that validate regulatory adherence **?**. The promise of this application lies in transforming traditionally manual, labor-intensive, and often error-prone compliance checks into more proactive, continuous, and scalable processes. By embedding AI-driven checks directly into development workflows, the aim is to foster "compliance by design" and "compliance by default," ensuring that software systems inherently meet specified standards from their inception, thereby reducing the need for costly retrospective remediation ??. This dimension is extensively explored in Sections 4 and 5 of this review, which detail core and advanced AI applications for compliance detection and proactive integration.

The second, equally critical dimension, addresses the need for compliance *of* AI systems themselves. As AI becomes increasingly integral to software development, the systems we build must inherently embody principles of trustworthiness, ethics, and sustainability. This dimension moves beyond merely applying AI to compliance tasks and instead scrutinizes the internal integrity, societal impact, and governance of the AI systems being developed. Key concerns here include ensuring fairness and non-discrimination, where AI models must not perpetuate or amplify societal biases; promoting transparency and explainability, allowing stakeholders to understand how AI decisions are made; safeguarding privacy and ensuring robust data governance for sensitive information processed by AI; and establishing clear accountability for AI system outcomes ??. Furthermore, the significant energy consumption and carbon footprint associated with training and deploying large AI models have emerged as a critical environmental sustainability compliance concern ??. Addressing these issues requires a paradigm shift from abstract ethical principles to concrete, verifiable claims and actionable engineering practices ?. This dimension necessitates the development of sociotechnical mechanisms, including institutional, software, and hardware solutions, to support external scrutiny, auditing, and accountability throughout the AI system development lifecycle ?. The challenges and frameworks for achieving compliance *of* AI systems are the primary focus of Section 3 and are further elaborated in Section 6, which discusses critical considerations for trustworthy AI-driven compliance systems.

In essence, "AI for Software Development Compliance" navigates a dual imperative: harnessing AI's transformative power to enhance the compliance posture of software products, while simultaneously ensuring that the AI systems themselves are constructed and operated in a manner that adheres to stringent ethical, regulatory, and societal standards. This review will explore the literature through this lens, examining both the technical advancements in AI-driven compliance automation and the critical frameworks for building trustworthy and responsible AI systems.

1.3 Scope and Structure of the Review

This literature review systematically delineates the intricate and rapidly evolving landscape of Artificial Intelligence (AI) in software engineering, with a particular emphasis on its profound implications for software development compliance. Its primary objective is to provide a comprehensive and critically analyzed overview, addressing the dual challenge of leveraging AI to enhance compliance processes while simultaneously ensuring that AI systems themselves are developed and deployed in a compliant, ethical, and trustworthy manner. This section outlines the review's boundaries, its pedagogical progression, and the thematic architecture designed to foster a coherent understanding of this interdisciplinary field.

The scope of this review is intentionally broad, encompassing foundational AI capabilities, advanced generative AI applications, and critical governance considerations across the entire Software Development Life Cycle (SDLC). Temporally, it focuses predominantly on contemporary advancements, particularly from the last decade, while also acknowledging seminal works that established key paradigms. Methodologically, the review synthesizes insights from empirical studies, systematic literature reviews, conceptual frameworks, and technical innovations, bridging theoretical discussions with practical applications. It explicitly addresses the recognized gap between abstract ethical principles for AI and their concrete operationalization in software engineering practice, as highlighted by works examining the challenges of integrating Responsible AI (RAI) throughout the SDLC **???**.

The review adopts a structured, pedagogical progression, moving from prerequisite knowledge to advanced applications and critical meta-concerns, ensuring that each section builds logically upon the preceding one.

1. **Introduction (Section 1)**: This initial section establishes the foundational context, defining the transformative impact of AI on software engineering and clarifying the multifaceted concept of "AI for Software Development Compliance." It sets the

stage by outlining the review's scope and structure, as presented here.

2. **Foundational Capabilities of AI in Software Engineering (Section 2)**: This section serves as a technical prerequisite, exploring the fundamental advancements that underpin AI's application in software development. It delves into the creation of large-scale datasets for code-related tasks, the empirical evidence of AI's impact on developer productivity, and the emergence of autonomous AI agents and platforms. Understanding these core capabilities is crucial for appreciating how AI can be effectively leveraged for compliance-specific applications, addressing the data and infrastructure aspects often overlooked in higher-level discussions **?**.

3. **The Imperative for Trustworthy and Responsible AI Development (Section 3)**: Shifting focus, this section addresses the critical need for AI systems themselves to be compliant and trustworthy. It examines conceptual frameworks for Trustworthy AI (TAI) and verifiable claims, ethical considerations, and maturity models for Responsible AI. This section is vital because, as empirical studies reveal, there is a significant gap between high-level ethical guidelines and their practical implementation in industry **???**. It highlights the necessity of integrating ethical and compliance considerations across the entire AI lifecycle, including often-neglected stages like documentation, monitoring, and risk assessment, particularly in regulated environments like fintech **?**. This proactive approach to AI compliance is essential to prevent risks to societal goals and ensure regulatory adherence **?**.

4. **Core AI Applications for Automated Compliance Detection (Section 4)**: Building on the foundational capabilities, this section explores the direct application of AI technologies to automate the detection of compliance rules. It covers traditional AI/ML techniques for code and artifact analysis, as well as specialized applications for security vulnerabilities and data privacy regulations. This section illustrates how AI moves from general software engineering tasks to specific compliance-checking functions.

5. **Advanced AI Applications for Proactive Compliance and Lifecycle Inte-**

gration (Section 5): This section advances the narrative from reactive detection to proactive compliance. It investigates how Large Language Models (LLMs) and Generative AI (GenAI) can interpret complex regulatory texts, map policies to code, and even generate inherently compliant software components. Furthermore, it examines the integration of these AI-driven mechanisms into modern DevOps and Agile workflows, emphasizing continuous compliance throughout the SDLC.

6. **Critical Considerations for Trustworthy AI-Driven Compliance Systems (Section 6)**: Following the exploration of AI's capabilities, this section delves into the meta-concerns essential for the successful and responsible deployment of AI-driven compliance systems. It addresses the need for Explainable AI (XAI) for auditing and trust, the role of blockchain for immutable evidence trails, and the crucial human factors that influence the integrity and auditability of AI-assisted compliance workflows. This section bridges technical solutions with practical, ethical, and regulatory demands, reinforcing the themes introduced in Section 3 by focusing on how to *trust* the AI systems performing compliance tasks.

7. **Conclusion and Future Directions (Section 7)**: The concluding section synthesizes key findings, identifies unresolved tensions, theoretical gaps, and practical challenges, and outlines promising future research directions. It underscores the ongoing need for interdisciplinary approaches and robust validation to achieve truly comprehensive and trustworthy AI-driven compliance.

This structured approach ensures a balanced perspective, encompassing both the technical prowess of AI in software engineering and the critical imperative for ethical, responsible, and compliant AI development and deployment. By progressing from foundational elements to advanced applications and then to overarching governance, the review aims to provide a holistic understanding of the intellectual trajectory and future challenges in "AI for Software Development Compliance." The emphasis throughout is on synthesizing connections and evolutionary trends, rather than merely cataloging individual studies, thereby offering a coherent narrative that addresses the complex interplay between AI innovation and regulatory necessity.

2 Foundational Capabilities of AI in Software Engineering

2.1 AI for Code: Datasets and Core Capabilities

The foundational advancement of Artificial Intelligence in software engineering is intrinsically linked to the development of robust data infrastructure and the establishment of core AI capabilities for understanding, processing, and manipulating code. This subsection delves into the essential data resources and initial AI systems that have addressed the data bottleneck, enabling the development of models capable of tasks such as code similarity, classification, and translation, thereby laying the groundwork for more sophisticated compliance-related analyses.

Historically, a significant impediment to AI for code research was the scarcity of large-scale, diverse, and richly annotated code datasets. This data bottleneck limited the training of sophisticated models capable of generalizing across various programming languages and tasks. A pivotal contribution to overcoming this challenge was the introduction of CodeNet by puri2021d34. CodeNet represents a monumental dataset, comprising over 14 million code samples and 500 million lines of code spanning 55 programming languages. This scale and linguistic diversity significantly surpassed its predecessors, such as POJ-104 and GCJ, which were often limited in scope or lacked comprehensive metadata. CodeNet's richness is further enhanced by crucial metadata, including submission status (e.g., Accepted, Wrong Answer), execution time, memory usage, problem descriptions, and input/output test cases. The methodologies employed for its creation involved rigorous data curation, meticulous cleansing using Jaccard similarity to identify and flag near-duplicates, and the provision of pre-processing tools for tokenization and parse-tree generation. These efforts ensured the high quality and usability of the data for training robust AI models. However, despite its scale, CodeNet primarily consists of solutions to competitive programming problems, which may introduce biases towards specific algorithmic patterns and pedagogical code, potentially limiting its direct applicability to complex enterprise-level software with diverse architectural patterns, proprietary APIs, and varied coding styles **?**. Furthermore, the provenance of such scraped data raises concerns regarding potential license contamination and the overall quality of code from public sources.

The availability of such large-scale datasets has been instrumental in enabling the development of foundational AI capabilities for code. These capabilities often begin with learning effective code representations. Models like CodeBERT **?** and GraphCodeBERT **?** (though not explicitly in the filtered list, these are seminal examples of such foundational models that leverage large code corpora) are pre-trained on vast amounts of code and natural language to learn rich, contextual embeddings of code. These learned representations underpin a variety of core tasks:

- **Code Similarity and Clone Detection**: Identifying functionally similar or duplicated code segments, crucial for refactoring, plagiarism detection, and identifying potential vulnerabilities.
- **Code Classification**: Categorizing code based on programming language, task type, or intent, which supports automated code organization and analysis.
- **Code Translation/Transpilation**: Converting code between different programming languages, a complex task that benefits from deep semantic understanding.
- **Code Summarization**: Generating concise natural language descriptions of code snippets, aiding documentation and code comprehension.
- **Code Search**: Enabling developers to find relevant code based on natural language queries, improving productivity.

Building upon these representational capabilities, initial generative AI models have emerged, aiming to automate code creation and assist developers. These tools leverage the patterns learned from extensive code datasets to generate code snippets, complete functions, and even suggest solutions to programming problems. For instance, generative AI algorithms are increasingly employed for automated test-case generation and bug identification, analyzing codebases and execution traces to uncover test scenarios and detect anomalous patterns indicative of bugs or vulnerabilities **?**. This capability promises amplified test coverage and efficiency gains, directly contributing to software quality.

However, the initial capabilities of AI for code, particularly in code generation, are not without significant limitations, necessitating critical human oversight. While these tools can generate functional code, they often struggle with accuracy and deep contextual understanding, frequently producing suboptimal, incorrect, or contextually irrelevant code **?**. A critical evaluation by pudari2023oep demonstrated that even prominent AI code completion tools largely fail to suggest idiomatic code or adhere to established best practices as their primary suggestions. This suggests that current models are heavily influenced by the frequency of patterns in their training data rather than their optimality or adherence to higher-level software quality attributes, requiring developers to critically review and often correct AI-generated output. Furthermore, the integration of generative AI into software development introduces security risks, as AI-generated code can inadvertently replicate insecure coding practices, introduce biases, or even "hallucinate" non-sensical or vulnerable code **?**. These issues underscore the need for robust security measures, thorough developer review, and continuous testing to mitigate the risks associated with AI-assisted code generation.

In conclusion, the foundational efforts in creating large-scale, diverse, and richly annotated datasets like CodeNet **?** have been instrumental in overcoming the data bottleneck for AI in code. These datasets have enabled the development of initial AI capabilities ranging from basic code understanding and representation to foundational tasks like code similarity, classification, translation, and initial code generation. While these advancements significantly enhance software development, the inherent limitations in the quality, security, and contextual understanding of AI-generated code **???** highlight the continuous need for improved data curation, more sophisticated AI models, and robust human oversight. These foundational capabilities are indispensable for building future AI systems capable of performing complex compliance-related analyses, such as automated security auditing, license compliance checking, and adherence to coding standards, by providing the underlying intelligence to understand and interact with code at a deep level.

2.2 AI-Assisted Development: Productivity and Human-AI Collaboration

The advent of artificial intelligence (AI) tools has initiated a transformative period in software development, profoundly impacting developer productivity and reshaping the dynamics of human-AI collaboration. This subsection explores the early empirical evidence of AI's influence on development speed, the evolving models of human-AI interaction, and the critical human factors that govern adoption, benefits, and challenges in this new paradigm.

Early empirical studies have consistently demonstrated significant productivity gains from AI-assisted coding tools. A seminal controlled experiment by peng2023uj3 provided the first rigorous evidence, showing that professional programmers using GitHub Copilot completed a standardized task 55.8

Beyond mere speed, research has begun to redefine the nature of human-AI interaction in software development, moving towards more collaborative models. alves2023ao6 introduced the conceptual framework of the "Centaur Programmer," drawing an analogy from advanced chess where human-AI teams outperform either humans or AI alone. This framework proposes novel collaboration models such as Guidance, Sketch, and Inverted Control, emphasizing a synergistic partnership rather than AI as a mere tool or replacement. Building on this vision, hassan2024hq8 and hassan2024pqx conceptualized "goal-driven AI pair programmers" and "AI-native Software Engineering (SE 3.0)," advocating for an intent-first, conversational development paradigm where AI acts as an intelligent, deeply SE-aware teammate. This shift in perspective is further articulated by meske2025khk, who defined "vibe coding" as a reconfiguration of intent mediation from deterministic instruction to probabilistic inference, fundamentally altering cognitive work and professional expertise.

Understanding the adoption dynamics and human factors is crucial for integrating these tools effectively. russo2023kua conducted a mixed-methods study, developing the "Human-AI Collaboration and Adaptation Framework (HACAF)" and empirically demonstrating that workflow compatibility is the predominant driver for early Generative AI adoption in software engineering, challenging traditional technology acceptance models. Deepening this understanding, li2024voc employed Socio-Technical Grounded Theory to develop a comprehensive "Theory of AI Tool Adoption," identifying intricate "push-pull" relationships between individual and organizational motives and challenges. Qualitative insights from pilot case studies, such as that by coutinho20245vb, further detail how professionals perceive and integrate generative AI tools into their daily work, highlighting both benefits and limitations. An empirical study by rasnayaka2024xtw on LLM usage in academic software engineering projects revealed that students primarily use LLMs for foundational code structures and syntax, with varying levels of human intervention required to integrate AI-generated code, suggesting that the quality and utility of AI outputs still necessitate human oversight.

However, the integration of AI tools is not without its challenges, particularly concerning collaboration, information seeking, and code quality. song20241ql found that while GitHub Copilot increased project-level code contributions in open-source software development, it also unexpectedly increased coordination time for code integration, especially for peripheral developers with less project familiarity. This highlights the complex social dynamics introduced by AI in team environments. Similarly, haque20246hg showed that while AI assistants enhance efficiency in information seeking, they introduce new challenges related to validating AI-generated information, often requiring developers to cross-reference multiple sources due to non-prescriptive language and inconsistent responses. A systematic literature review by sergeyuk2025bfj on Human-AI Experience in IDEs further synthesizes these impacts, noting concerns about over-reliance and the need for better validation mechanisms.

A critical challenge lies in ensuring the security and quality of AI-assisted development. A pivotal empirical study by perry2022cq5 demonstrated that developers using AI assistants wrote significantly less secure code and exhibited increased overconfidence in its security, revealing a direct and severe security compliance risk. This finding is corroborated by klemmer20246zk, who, through a qualitative study, found that developers generally mistrust the security of AI suggestions but still widely use them for security-critical tasks, necessitating rigorous manual review akin to human-generated code. These security concerns are part of broader ethical considerations, including "hallucinations," the "black box" problem, and intellectual property risks, as highlighted by parikh2023x5m in their systematic review of Generative AI in software product management. Addressing these quality and security limitations is paramount. In response, technical solutions are emerging, such as "Copilot for Testing" proposed by wang2025vty, which leverages a context-based Retrieval Augmented Generation (RAG) module to dynamically improve bug detection accuracy and test coverage, thereby enhancing the quality of AI-assisted code.

In conclusion, the literature reveals a dual trajectory for AI-assisted development: significant productivity gains are evident, yet these benefits are intertwined with complex human factors, evolving collaboration models, and critical challenges related to code quality, security, and the need for continuous human oversight. The journey from AI as a mere productivity tool to a collaborative partner, as envisioned by the "Centaur Programmer" and "AI-native SE" paradigms, necessitates a deeper understanding of human-AI interaction. Unresolved issues include designing AI systems that inherently mitigate security risks, fostering effective human-AI communication for goal alignment, and developing robust validation mechanisms for AI-generated content. Future research must focus on designing effective and compliant AI-driven development workflows that strategically leverage human strengths while actively mitigating AI limitations, ensuring both efficiency and trustworthiness in the evolving software landscape.

2.3 Autonomous AI Agents and Platforms for Software Engineering

The landscape of software engineering is being reshaped by the emergence of autonomous AI agents and sophisticated platforms designed to perform complex development tasks with increasing independence. This shift moves beyond mere AI assistance towards systems capable of understanding, planning, executing, and verifying software changes, paving the way for more autonomous compliance-related tasks.

The conceptual foundation for evaluating generalist agents was laid by early work such as the Arcade Learning Environment (ALE) **?**, which provided a diverse platform for assessing AI algorithms across numerous tasks, emphasizing the need for domain-independent competency. Complementing this, the development of large-scale, diverse datasets like CodeNet **?** became crucial, offering over 14 million code samples in 55 programming languages with rich metadata, providing the necessary raw material for training AI models capable of understanding and generating code.

Recent advancements have focused on empowering Large Language Models (LLMs) to not only use but also create their own tools, significantly enhancing their autonomy. The LLMs AsToolMakers (LATM) framework **?** exemplifies this by introducing a two-phase process where a powerful LLM acts as a "tool maker" to craft reusable Python functions from demonstrations, while a lightweight LLM serves as a "tool user" to apply these verified tools. This innovative approach, coupled with a functional cache, optimizes cost and performance by strategically dividing labor and allowing agents to extend their capabilities dynamically.

Building upon these foundational capabilities, platforms are emerging to provide comprehensive environments for generalist AI agents. OpenHands **?** stands out as an open platform designed for AI software developers, offering a secure Docker-sandboxed runtime environment for each task session. This platform provides a comprehensive, programming language-based action space, allowing agents to interact with operating systems via arbitrary Python code (`IPythonRunCellAction`), execute bash commands (`CmdRunAction`), and browse the web using a domain-specific language (`BrowserInteractiveAction`).

OpenHands further enhances agent capabilities through an extensible `AgentSkills` library for specialized tools and a multi-agent delegation mechanism, providing an "ideal interface" for agents to engage in complex software-related tasks.

Beyond generalist platforms, specialized agents are demonstrating advanced capabilities in specific software engineering domains. MarsCode Agent **?**, for instance, focuses on AI-native automated bug fixing through a novel multi-agent collaborative framework. This agent integrates LLMs with advanced code analysis techniques like Code Knowledge Graphs (CKG) and Language Server Protocols (LSP) for deep code understanding across large codebases, utilizing a containerized sandbox for dynamic debugging and accurate patch generation. MarsCode Agent represents a significant step in overcoming LLM limitations in handling real-world bug fixing complexity by structuring the process into distinct agent roles with specialized toolsets.

As these autonomous agents become more sophisticated, rigorous evaluation of their generalization capabilities becomes paramount. SWE-bench Multimodal (SWE-bench M) **?** addresses this critical need by introducing a novel benchmark dataset that evaluates AI systems on visual, user-facing JavaScript software issues. Unlike prior text-based, Python-centric benchmarks, SWE-bench M explicitly requires multimodal reasoning to interpret images and videos in problem statements, alongside cross-language generalization to JavaScript. Evaluations on SWE-bench M have revealed significant limitations in existing AI systems, with even adapted agents like `SWE-agent M` achieving modest success rates, underscoring the substantial research required for AI systems to generalize across diverse languages and visual domains.

In conclusion, the progression from foundational general agent evaluation and large-scale code datasets to platforms enabling autonomous tool creation and comprehensive interaction environments marks a significant leap in AI for software engineering. While specialized agents like MarsCode Agent demonstrate impressive capabilities in targeted tasks, benchmarks like SWE-bench Multimodal highlight the persistent challenge of achieving true generalization across complex, multimodal, and multi-language software domains. Bridging this gap remains a key unresolved issue, requiring future research to integrate advanced LLM reasoning with robust, language-agnostic tools and multimodal perception, which is essential for these agents to reliably perform autonomous compliance-related tasks in real-world software development.

3 The Imperative for Trustworthy and Responsible AI Development

3.1 Conceptualizing Trustworthy AI and Verifiable Claims

The imperative for Trustworthy AI (TAI) marks a crucial evolution in AI development, demanding a rigorous shift from abstract ethical principles to concrete, verifiable claims about AI systems' behavior, safety, security, fairness, and privacy. This transition is fundamental for establishing robust regulatory compliance, fostering public confidence, and ensuring accountability in AI technologies. The literature emphasizes the development of comprehensive sociotechnical mechanisms—encompassing institutional, software, and hardware solutions—that enable external scrutiny and demonstrable trustworthiness, thereby bridging the gap between high-level ideals and actionable, auditable practices.

A foundational contribution to this paradigm is presented by **?**, which proposes a comprehensive "toolbox" of mechanisms for supporting verifiable claims in AI development. This work moves beyond generic ethical guidelines, advocating for the necessity of falsifiable statements about AI systems and their development processes, backed by tangible evidence. The proposed framework integrates institutional mechanisms such as third-party auditing, red teaming exercises, and the sharing of AI incident reports to foster external pressure and transparency. Crucially, it also advocates for software-level innovations, including detailed audit trails to capture the AI lifecycle information for safety-critical applications, and interpretability methods explicitly designed to support risk assessment and auditing. Furthermore, **?** emphasizes hardware-level solutions, such as secure hardware features for machine learning accelerators and high-precision compute measurement, to enhance the verifiability of privacy, security, and resource usage claims.

While **?** provides a robust conceptual blueprint, its primary limitation lies in its theoretical nature, necessitating empirical validation and practical operationalization of these proposed mechanisms.

The conceptualization of TAI, however, extends beyond a single framework, encompassing diverse governance approaches and highlighting significant practical challenges in implementation. **?**'s rapid review of Responsible AI (RAI) frameworks reveals a critical gap: despite the proliferation of ethical guidelines, only a small fraction offer practical tools, and most focus heavily on early development phases, neglecting design, development, testing, and deployment. This underscores the difficulty in translating abstract principles into actionable engineering practices. Complementary to this, **?** advocates for a human rights-centered design approach, arguing that international human rights standards offer a universally recognized basis for AI governance, requiring systemic consideration at every stage of design and deployment, supported by technical verification and independent oversight. Similarly, **?** proposes algorithmic auditing as a regulatory requirement to verify compliance, particularly in sensitive domains like financial inclusion, highlighting the need for external accountability beyond self-regulation. These diverse perspectives collectively emphasize that verifiable claims must be embedded within a broader, legally mandated, and continuously audited governance structure.

Empirical studies further underscore the practical difficulties in achieving trustworthiness, revealing a significant gap between ethical aspirations and real-world development practices. **?** found that ethical considerations are often ignored in agile, startup-like environments, where a "prototype" mindset defers ethical scrutiny. This is corroborated by **?**, which identified a notable gap between AI ethics guidelines and actual company practices, particularly concerning societal well-being, diversity, and fairness. **?**'s grounded theory review of practitioners' views on AI ethics further details these challenges, categorizing them into awareness, perception, need, and approach, revealing that while practitioners acknowledge ethical concerns, translating them into concrete actions remains problematic. To bridge this gap, **?** proposes operationalized software engineering patterns derived from empirical studies, aiming to integrate responsible AI considerations throughout the entire

AI system lifecycle, moving beyond abstract principles to concrete, verifiable specifications and continuous monitoring.

The necessity for verifiable claims is particularly acute when considering the inherent risks of AI systems. **?** identifies various AI-specific sources of risk, including those stemming from modern machine learning methods, which necessitate adapted risk management strategies. In the realm of security, **?** empirically demonstrated that developers using AI code assistants often produce significantly less secure code and exhibit increased overconfidence, highlighting a critical need for verifiable security claims and robust audit trails for AI-assisted development processes. This concern is amplified by the emergence of autonomous AI agents, whose unpredictability, complexity, and interactions with untrusted external entities pose significant security challenges that demand rigorous verification mechanisms **?**. Beyond security, environmental sustainability has emerged as a critical dimension requiring verifiable claims. The holistic framework by **?** characterizes AI's environmental implications across its lifecycle, while **?** introduces methods for measuring real-time carbon intensity. These advancements directly operationalize the "high-precision compute measurement" advocated by **?**, providing the granular data necessary for making verifiable claims about the environmental impact of AI workloads.

In conclusion, the conceptualization of Trustworthy AI has evolved from a normative ideal to a demand for demonstrable and auditable practices. While foundational frameworks like **?** provide a crucial blueprint for verifiable claims through institutional, software, and hardware mechanisms, empirical evidence consistently reveals significant challenges in their practical implementation across the software development lifecycle **???**. The diverse risks posed by AI, from security vulnerabilities in AI-assisted coding to the environmental footprint of AI workloads **?????**, reinforce the urgency for robust sociotechnical solutions. The ongoing challenge lies in fully integrating these mechanisms into AI development, ensuring that trustworthiness is not merely an aspiration but an actionable, auditable, and continuously verified practice, thereby establishing the bedrock for effective regulatory compliance and enduring public confidence.

3.2 Ethical Considerations and Maturity Models in AI Development

The escalating deployment of Artificial Intelligence (AI) systems across diverse sectors has illuminated a critical challenge: the persistent gap between abstract ethical principles and their concrete operationalization within the AI software development lifecycle (SDLC). While numerous high-level ethical guidelines and frameworks have been proposed by governmental bodies and academic institutions, empirical evidence consistently reveals that these principles are often overlooked or deferred in practice. For instance, **?** conducted a multiple case study in startup-like environments, uncovering a pervasive "prototype" mindset that frequently served as a justification for deferring ethical considerations, leading to their complete neglect in practical AI development. This finding was further substantiated by **?**, whose gap analysis across 39 companies confirmed a notable disconnect between AI ethics guidelines and industry practice, particularly concerning novel requirements for societal well-being and fairness. Such empirical insights underscore the urgent need to translate theoretical ethical imperatives into actionable engineering practices.

To bridge this crucial research-practice divide, there is a clear and pressing demand for structured approaches to operationalize Responsible AI (RAI). A significant direction in this regard involves the development of AI ethics maturity models, which offer systematic frameworks for organizations to assess, benchmark, and incrementally enhance their ethical AI development processes. **?** cogently argues for the immediate necessity of an AI (Ethics) Maturity Model, explaining that traditional software engineering maturity models (e.g., CMMI, SPICE) are insufficient. This inadequacy stems from AI's unique characteristics, such as its probabilistic nature, inherent data-centricity, and evolving quality attributes like fairness, trustworthiness, and transparency, which demand distinct considerations not adequately addressed by existing models.

Building upon this identified need, **?** proposed and statistically assessed AI-MM, a maturity model specifically designed for trustworthy AI software development. This model integrates common AI processes with fairness-specific considerations within a SPICE-like framework, providing a structured approach to measure maturity levels and offer practical guidelines for enhancement. Its effectiveness was demonstrated through application to 13 real-world AI projects, showcasing its utility in identifying areas for improvement. While AI-MM offers a valuable, process-oriented assessment tool, its primary focus on fairness, though critical, highlights a potential limitation in its comprehensive coverage of the broader spectrum of ethical principles (e.g., accountability, privacy, transparency) that ? advocates for. The challenge remains in developing models that can systematically assess and guide improvement across all facets of RAI.

Beyond maturity models, comprehensive roadmaps and operationalized patterns offer more granular, actionable guidance for embedding ethical principles across all phases of the SDLC. ? contributed to this by conducting a grounded theory literature review that synthesized AI practitioners' views, challenges, and approaches to ethics, resulting in a taxonomy that illuminates the human element in ethical AI development. This taxonomy provides crucial insights into the practical realities faced by developers, highlighting the need for solutions that resonate with their workflows and concerns. Complementing this, ? presented an empirical study with AI scientists and engineers, which informed the development of a novel template for operationalizing abstract AI ethics principles into concrete software engineering patterns. These patterns, encompassing both process and design aspects, aim to integrate responsible AI considerations throughout the entire AI system lifecycle, from requirements engineering to deployment and operation. Further extending this, ? developed a comprehensive roadmap for Software Engineering for Responsible AI, derived from a systematic literature review. This roadmap advocates for a holistic, process-oriented approach, detailing multi-level governance strategies, lifecycle-integrated practices, and Responsible-AI-by-Design principles to move beyond isolated algorithmic solutions. ? further reinforces this by providing empirical insights into adapting existing software engineering processes for implementing responsible AI, covering aspects from requirements engineering to deployment.

Synthesizing these approaches, maturity models like AI-MM (?) provide a framework for *assessing* an organization's current state and guiding incremental improvements, akin to a diagnostic tool. In contrast, the roadmaps and operational patterns proposed by ?

and **?** offer the *prescriptive guidance*—the "how-to"—for achieving higher maturity levels by integrating ethical considerations directly into development practices. The taxonomy by **?** provides the critical *context* of practitioner perspectives, essential for designing effective and adoptable solutions. The collective aim is to foster a shift from reactive ethical audits to proactive, "ethics-by-design" principles, ensuring that AI systems are developed responsibly from inception, encompassing fairness, transparency, and accountability as intrinsic quality attributes.

In conclusion, the literature reveals a concerted and evolving effort to move beyond abstract ethical pronouncements towards a holistic, process-oriented approach for ethical compliance in AI development. While significant progress has been made in proposing conceptual frameworks, maturity models, and comprehensive roadmaps, the persistent empirical evidence of ethical oversights (**?**, **?**) indicates that the challenge of widespread adoption and effective integration remains substantial. Future research must therefore focus on several critical areas: conducting longitudinal studies to validate the long-term effectiveness and scalability of proposed maturity models and roadmaps in diverse organizational contexts; developing more sophisticated tools that not only guide but also *enforce* ethical practices within the SDLC; and exploring the socio-technical factors that impede or facilitate the integration of these frameworks into existing agile and DevOps workflows. Bridging the gap between theoretical ethical principles and practical, ingrained development processes requires continuous refinement of these structured approaches and fostering interdisciplinary collaboration to ensure AI systems are developed responsibly and ethically from their foundational design.

3.3 Environmental Sustainability of AI Systems

The escalating energy consumption and carbon footprint of Artificial Intelligence (AI) systems pose a significant environmental challenge, necessitating a dedicated focus on sustainability within AI compliance frameworks. Addressing this concern requires comprehensive methodologies for quantifying environmental impact, coupled with architectural and design strategies for building 'green AI' systems.

Early research highlighted the critical need for a holistic understanding of AI's environmental implications. **?** introduced a foundational framework for characterizing AI's carbon footprint, moving beyond isolated model training costs to encompass the entire Machine Learning (ML) development cycle—from data processing and experimentation to training and inference—and the full life cycle of AI system hardware. This work empirically demonstrated that embodied carbon from hardware manufacturing can be a dominating factor, especially when operational emissions are mitigated by carbon-free energy, emphasizing the importance of hardware-software co-design for substantial reductions.

Building upon the imperative for comprehensive quantification, subsequent work focused on developing more granular and actionable measurement tools. **?** presented a practical framework for measuring Software Carbon Intensity (SCI) for AI workloads in cloud instances, uniquely leveraging real-time, location-based, and time-specific marginal emissions data. This approach allows for more informed decision-making, identifying that choosing optimal geographic regions and even the time of day for computation can significantly reduce operational carbon emissions, thereby empowering ML practitioners to make carbon-aware choices.

While quantifying the problem is crucial, the field has progressed towards embedding sustainability directly into the AI system design and development process. **?** proposed an architecture-centric approach, GAISSA, to integrate "greenability" as a first-class concern. This initiative introduces an AI-specific quality model for greenability, predictive models to guide sustainability-aware AI model training, and a catalogue of architecture and design patterns specifically tailored for building green AI-based systems. This moves beyond post-hoc measurement to proactive design, addressing the lack of concrete, actionable methods for implementing energy efficiency throughout the AI engineering lifecycle.

To further operationalize and assess these green practices, metrics are being developed to quantify the adoption of sustainable AI. **?** introduced the "Green AI Quotient" (GAQ) as a novel metric designed to assess the "greenness" of any AI-based system and its development process. This metric aims to bridge the gap between sustainable AI research and its industry-scale adoption, encouraging the integration of energy-efficient AI research and practices into real-world projects.

In summary, the literature demonstrates a clear progression from identifying and holistically quantifying the significant environmental impact of AI systems ?, to developing granular, real-time measurement tools for cloud environments ?, and finally to proposing systematic, architecture-centric design methodologies and metrics for building inherently sustainable AI systems ??. Despite these advancements, challenges remain in standardizing metrics, ensuring widespread adoption of green AI practices across diverse industries, and continuously integrating these considerations into the rapidly evolving AI software development lifecycle to meet broader Environmental, Social, and Governance (ESG) compliance requirements. Future research must focus on developing automated tools and robust frameworks that seamlessly embed greenability from conception through deployment and operation, fostering a truly sustainable AI ecosystem.

4 Core AI Applications for Automated Compliance Detection

4.1 Traditional AI/ML for Code and Artifact Analysis

Automated compliance checking is a cornerstone for ensuring software adheres to established rules, standards, and regulatory requirements throughout the software development lifecycle. This subsection explores the application of traditional Artificial Intelligence (AI) and Machine Learning (ML) techniques, including Natural Language Processing (NLP) and static analysis, to systematically analyze software requirements, design documents, and source code. The objective is to identify deviations from established norms, thereby reducing manual effort, improving accuracy, and streamlining early-stage compliance verification through reactive detection of implicit rule violations and explicit checking against formalized knowledge.

Early and ongoing efforts in leveraging AI for software engineering, particularly in the requirements phase, have established foundational groundwork for compliance verification.

Natural Language Processing (NLP) techniques are extensively used to analyze textual requirements documents for ambiguities, inconsistencies, or potential non-compliance. **?** provides a comprehensive review of AI in Software Requirements Engineering (RE), highlighting the significant role of NLP and supervised learning techniques like classification in analyzing requirements. This approach is crucial for identifying implicit rules or making them explicit and detectable, serving as an initial step towards automated compliance checking. Similarly, **?** further emphasizes NLP's role in automating requirements classification and sentiment analysis, which streamlines RE practices and can help flag compliance-critical requirements for closer scrutiny. However, the complexity of requirements for AI-based systems themselves presents unique challenges, as highlighted by **?**. Issues such as the "black-box" nature of ML models and their inherent data dependencies complicate traditional RE activities, impacting the traceability and auditability of requirements for AI systems used in compliance, thereby posing a meta-challenge for trustworthy AI-driven compliance.

Beyond requirements, traditional AI/ML techniques extend to the analysis of design documents and architectural artifacts. Knowledge-based systems, often relying on formalized ontologies and rule engines, enable explicit checking against predefined compliance rules. These systems can represent regulatory text and software artifacts in a structured manner, allowing for logical inference to identify non-compliant designs or architectural structures. More recently, AI, including advanced NLP capabilities, has been applied to semi-automatically generate software architectures from textual requirements. For instance, **?** proposes an iterative, AI-based process that leverages Large Language Models (LLMs) to generate initial domain models and architecture candidates from natural language requirements. While utilizing modern LLMs, the underlying task of translating natural language into structured design artifacts and evaluating them against quality attributes (which can include compliance) represents an evolution of traditional AI's role in design analysis, aiming to reduce manual effort and explore more design alternatives.

For source code and other executable artifacts, Machine Learning (ML) classifiers and pattern mining techniques are widely employed. ML classifiers, trained on labeled datasets of compliant and non-compliant code snippets or design patterns, can automatically categorize new artifacts, significantly reducing manual effort in compliance audits. For example, in the domain of software quality and security (which are often compliance concerns), AI-based defect prediction (SDP) frameworks like DePaaS, as discussed by **?**, utilize various ML models to isolate defective software modules and identify risky code changes. This directly contributes to compliance by flagging code that deviates from quality standards. **?** further corroborates this, noting the extensive adoption of ML and deep learning algorithms in development and testing phases for "defect prediction, code recommendation, and vulnerability detection initiatives." These initiatives often involve analyzing code features (e.g., from Abstract Syntax Trees, control flow graphs) to detect patterns indicative of security vulnerabilities or violations of secure coding standards. Similarly, API usage pattern mining employs data mining techniques to learn common or "correct" sequences of API calls from existing codebases. Deviations from these learned patterns can signal potential non-compliance with coding standards, security policies, or resource management rules, offering a reactive detection mechanism for implicit rule violations. Furthermore, AI-based software testing, as reviewed by **?**, enhances traditional testing methodologies to ensure quality, reliability, and security, directly contributing to compliance verification in the later stages of the development lifecycle.

Despite their utility, traditional AI/ML approaches face several limitations. Their effectiveness often hinges on the quality and quantity of labeled data, which can be expensive and time-consuming to acquire for specific compliance domains. The explicit formalization of complex compliance rules into knowledge-based systems or training data for ML models can also be challenging. Scalability can be an issue with complex, evolving regulations, requiring continuous model retraining and rule updates. Moreover, the interpretability of some ML models, often referred to as the "black-box" problem, can hinder auditability and trust in highly regulated environments, making it difficult to justify compliance decisions to human auditors or regulatory bodies. This challenge is particularly acute when the AI system itself is subject to compliance, as discussed by **?**. Nevertheless, these techniques have significantly advanced the automation of compliance checking, moving the field from purely manual reviews to more efficient, data-driven, and reactive verification processes, laying the groundwork for more proactive and advanced AI applications.

4.2 AI for Security and Vulnerability Compliance

Leveraging Artificial Intelligence (AI) has become indispensable in enhancing software security and ensuring compliance with increasingly stringent security policies and regulatory frameworks. This subsection delves into AI-driven approaches for proactively identifying, predicting, and mitigating security risks throughout the software development lifecycle, explicitly linking these capabilities to the verification of compliance against established standards and policies.

The foundation for effective AI-driven security analysis relies on extensive and diverse datasets. Large-scale repositories like CodeNet **?**, comprising millions of code samples across numerous programming languages with rich metadata, serve as crucial training grounds for AI models. These datasets enable the development of AI systems capable of understanding code semantics, identifying patterns, and ultimately detecting security flaws that violate coding standards or security policies. Building upon such foundations, AI has significantly advanced defect prediction. While traditional methods relied on defect density, modern approaches increasingly utilize machine learning for more granular and accurate predictions, even in large-scale software systems with hundreds of millions of lines of code **?**. Generative AI algorithms, for instance, are now being explored for automated test-case generation and bug identification, scrutinizing codebases and execution traces to detect coding mistakes, anomalous patterns, and potential security vulnerabilities **?**, thereby contributing to compliance with quality and security requirements.

More specifically, AI-driven techniques for vulnerability detection span various stages of the software development process, directly supporting compliance efforts. In Static Application Security Testing (SAST), deep learning models are employed to analyze source code without execution, identifying common weaknesses (CWEs) and insecure coding patterns that contravene organizational security policies or industry best practices like the

OWASP Top 10. Advanced approaches include the use of Graph Neural Networks (GNNs) to analyze Code Property Graphs (CPGs). GNNs are particularly effective as they can model complex data and control flows represented in CPGs, enabling the detection of sink-source vulnerabilities that span multiple functions and are often missed by simpler pattern-matching approaches, thus enhancing the ability to verify adherence to secure coding guidelines. Transformer models, often fine-tuned on vast code corpora, are also emerging as powerful tools for identifying insecure code snippets and suggesting secure alternatives, directly aiding developers in writing compliant code. While Large Language Models (LLMs) are increasingly applied to vulnerability detection, their practical utility for industry use is still under scrutiny. A qualitative study by Kholoosi et al. **?** found that while security practitioners perceive LLMs like ChatGPT as beneficial for tasks like vulnerability detection and penetration testing, the actual outputs are often generic and may not be appropriate for real-world industrial application, highlighting a gap between perception and the precision required for verifiable compliance.

Beyond static analysis, AI is crucial for real-time threat detection and integration into modern development workflows, directly supporting continuous compliance. Saleh et al. **?** present a concrete AI-driven solution for real-time anomaly detection in Continuous Integration/Continuous Deployment (CI/CD) pipelines within cloud environments. Their hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model achieves high accuracy (e.g., 98.69

It is important to acknowledge that while AI offers powerful detection capabilities, the increasing use of AI assistants in software development can inadvertently introduce new vulnerabilities. This necessitates even more robust AI-driven detection tools to identify flaws regardless of their origin, ensuring that the benefits of AI-assisted coding are maximized while associated security risks are minimized and compliance is maintained.

In conclusion, AI offers transformative potential for enhancing software security and compliance by providing advanced tools for vulnerability detection, real-time anomaly identification, and seamless integration into DevSecOps pipelines. These AI-driven approaches enable automated checking against specific security policies, standards like CWEs and OWASP, and regulatory frameworks. The literature highlights a dual imperative: developing more sophisticated AI techniques to detect vulnerabilities, including those potentially introduced by AI itself, and ensuring these detection systems provide verifiable evidence for compliance auditing. Future research must focus on bridging the gap between AI's analytical capabilities and the need for verifiable compliance, demanding robust, AI-driven evaluation frameworks and continuous refinement of AI-assisted secure development practices that are explicitly mapped to regulatory requirements.

4.3 AI for Data Privacy and Regulatory Adherence

Ensuring adherence to complex data privacy regulations and legal mandates, such as the General Data Protection Regulation (GDPR) **?** and the California Consumer Privacy Act (CCPA), presents a significant challenge in software development. These regulations impose stringent requirements on data handling, necessitating principles like data minimization, purpose limitation, and robust security measures. Traditionally, compliance has relied heavily on manual effort and specialized legal expertise, leading to high costs, inconsistencies, and a reactive approach to violations. Artificial intelligence offers transformative potential to automate the *detection* of data privacy violations in software artifacts and to facilitate the *interpretation* of regulatory texts to define actionable compliance checks, thereby embedding privacy considerations more effectively into software engineering practices.

Early efforts in AI-driven compliance focused on establishing foundational frameworks and leveraging traditional machine learning (ML) and Natural Language Processing (NLP) for initial detection. **?** and **?** proposed high-level, AI-driven frameworks for continuous compliance checking across the entire Software Development Life Cycle (SDLC), utilizing ML and NLP for early non-compliance detection. Building on this, **?** demonstrated a concrete application, employing supervised machine learning techniques such as Support Vector Machines and Naive Bayes to classify software requirements as compliant or non-compliant. This work validated the feasibility of ML for basic compliance assessments by identifying explicit rule violations in textual requirements. Similarly, **?** surveyed the application of AI, particularly NLP and classification, in Requirements Engineering (RE) to improve requirement quality, a methodology directly applicable to identifying privacy-related requirements and detecting their absence or ambiguity. While these foundational works laid the conceptual groundwork for automated detection, they often focused on less complex artifacts or lacked deep technical validation against real-world, intricate privacy regulations.

More advanced AI techniques, particularly deep learning, have been applied to directly detect data privacy violations within source code. For instance, **?** proposed deep learning models, specifically Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTMs), for detecting data privacy violations directly in source code. Their models were trained to identify code patterns indicative of non-compliance with regulations like GDPR and CCPA, addressing specific articles related to data handling and user consent. This approach moves beyond high-level document analysis to pinpoint actual implementation flaws, offering a more granular and precise detection capability. However, the effectiveness of such models heavily relies on the quality and representativeness of the training data, and they can struggle with novel attack vectors or highly obfuscated code.

A critical aspect of regulatory adherence is the accurate interpretation of complex legal texts and their translation into technical specifications for detection. The GDPR, as a prime example **?**, mandates principles like data minimization and privacy by design, which require careful interpretation to translate into verifiable code-level checks. While the proactive generation of compliant code is discussed elsewhere, AI can assist in *interpreting* these regulations to *derive* the rules and patterns necessary for detection. **?** demonstrated how Large Language Models (LLMs) can interpret complex, natural language regulatory texts and translate them into actionable software requirements. In the context of detection, this capability is invaluable for automatically extracting compliance rules, identifying relevant data types, and mapping legal obligations to specific code constructs or architectural patterns that an AI detection system can then monitor. This helps bridge the semantic gap between legal and technical domains, enabling the creation of more comprehensive and up-to-date detection rules.

As AI-driven systems become integral to privacy compliance, ensuring the privacy and trustworthiness of these AI tools themselves is paramount. The development of AI models often requires vast amounts of data, which can raise privacy concerns if sensitive code or user data is used for training. **?** proposed a decentralized governance framework for open-source AI-based Software Engineering (SE) tools centered on Federated Learning (FL). This approach enables multiple entities to collaboratively train and maintain AI code models, including those for privacy compliance detection, without directly sharing their proprietary local data, thereby preserving privacy during the development of the AI tool itself. Furthermore, the deployed AI models used for compliance checks must also be robust against privacy attacks. **?** introduced `SafeCompress`, a bi-objective optimized model compression framework that simultaneously optimizes model performance and safety, specifically addressing privacy leakage (e.g., membership inference attacks) in deep learning models. This ensures that the AI models embedded within software, which might handle sensitive data, are themselves privacy-preserving, aligning with the broader goal of trustworthy AI development **??**. The practical integration of ethical considerations, including data privacy, into AI development workflows remains a challenge, as highlighted by gap analyses showing discrepancies between ethical guidelines and industrial practice **?**.

Despite these advancements, challenges persist in AI-driven data privacy detection. Current methods often struggle with the dynamic nature of regulations, the ambiguity inherent in legal language, and the scalability required for large, complex software systems. False positives and negatives remain a concern, requiring significant human oversight. Moreover, while security is foundational for privacy (e.g., GDPR Article 32 mandates "security of processing"), general security compliance tools, such as those for anomaly detection in CI/CD pipelines **?**, must be explicitly linked to specific privacy principles to be considered direct privacy compliance measures. The "black box" nature of some advanced AI models also complicates auditing and justification of detected violations, posing a barrier to widespread adoption in highly regulated environments.

5 Advanced AI Applications for Proactive Compliance and Lifecycle Integration

5.1 Leveraging Large Language Models for Regulatory Interpretation and Policy-to-Code Mapping

The increasing complexity of regulatory landscapes presents a significant challenge for software development, often leading to a substantial gap between legal mandates expressed in natural language and their technical implementation as code-level policies. Bridging this divide necessitates sophisticated tools capable of interpreting nuanced regulatory texts and translating them into actionable software requirements and verifiable code. This subsection explores the transformative role of Large Language Models (LLMs) in addressing this challenge, moving beyond traditional keyword matching to enable semantic reasoning for automated policy-to-code compliance mapping.

Early efforts to automate regulatory compliance leveraged traditional Natural Language Processing (NLP) techniques to extract rules and requirements. For instance, **?** demonstrated how NLP could be utilized to analyze natural language regulatory documents, identifying and extracting compliance rules that could then be translated into software requirements. This approach was foundational in bridging the initial gap between legal text and technical specifications, laying the groundwork for more advanced semantic understanding. However, traditional NLP often struggled with the inherent ambiguity, context-dependency, and vastness of legal texts, limiting its ability to achieve deep contextual understanding and robust interpretation.

The advent of Large Language Models (LLMs) has marked a significant leap forward in this domain, offering unparalleled capabilities for understanding and generating human language. **?** showcased the transformative potential of LLMs by demonstrating their ability to interpret complex, natural language regulatory texts and subsequently translate them into actionable software requirements. This work highlighted how LLMs could move beyond superficial keyword matching to grasp the deeper contextual meaning of regulations, thereby enabling a more accurate and comprehensive integration of legal mandates into software engineering processes. Building on this enhanced interpretative power, **?** further advanced the application of LLMs specifically for automated policy-to-code compliance mapping. Their research illustrated how LLMs could directly map high-level compliance policies to concrete code implementations, effectively generating code-level policies or checks based on the nuanced understanding derived from regulatory input. This capability streamlines the integration of legal mandates into technical specifications, reducing manual effort and potential for human error.

Beyond mere interpretation and mapping, LLMs are also being leveraged for automated compliance assessment and proactive design. **?** explored the utility of LLMs for automated software compliance assessment, emphasizing their capacity to understand nuanced regulatory language and provide explanations for compliance decisions. This not only enhances the accuracy of assessments but also contributes to the crucial aspect of auditability in regulated environments. Extending this proactive dimension, **?** proposed the use of generative AI, often powered by LLMs, to proactively design software components that are inherently compliant from the outset. This paradigm shift aims to embed compliance "left" in the software development lifecycle, moving from reactive detection of non-compliance to preventative design, thereby minimizing rework and ensuring compliance by design. Similarly, **?** explored AI-assisted generation of compliance-aware code, where AI actively guides developers or auto-generates code snippets that inherently adhere to compliance rules, further solidifying the shift towards proactive compliance.

Despite these significant advancements, several challenges remain. The inherent "black box" nature of many LLMs can hinder explainability and auditability, which are paramount in highly regulated industries. While some research like **?** touches upon explanations, ensuring these are legally sufficient and contextually accurate remains an active area of research. Furthermore, the potential for LLM "hallucinations" or misinterpretations of complex legal jargon necessitates robust validation mechanisms and human-in-the-loop oversight. The computational cost associated with training and deploying large-scale LLMs, as well as the need for continuous adaptation to evolving regulations, also pose practical challenges. Future research must focus on developing more transparent and auditable LLM-based systems, integrating robust validation frameworks, and exploring efficient methods for continuous learning and adaptation to ensure the trustworthiness and scalability of automated policy-to-code mapping solutions.

5.2 Generative AI for Compliance-Aware Design and Code Generation

The increasing complexity of regulatory landscapes necessitates a paradigm shift from reactive compliance detection to proactive compliance-by-design, where adherence to rules and policies is embedded from the initial stages of software development. This subsection explores the innovative application of Generative AI (GenAI) to achieve this goal, focusing on approaches where AI actively guides developers or auto-generates design artifacts and code snippets that inherently adhere to compliance rules, security policies, or architectural guidelines. This proactive stance aims to prevent non-compliance from the outset, rather than merely detecting it post-factum, driven by stringent regulations such as the General Data Protection Regulation (GDPR) **?**.

Foundational work in AI for software engineering has broadly categorized AI's potential across the Software Development Life Cycle (SDLC) **?**. Building upon this, recent reviews specifically on Generative AI in software architecture underscore its application in architectural decision support, reconstruction, and the transformation from requirements to architecture **?**. However, these reviews also point to a critical gap in rigorous evaluation methodologies for GenAI outputs in architectural contexts, a limitation that becomes even more pronounced when considering compliance. Addressing this, eisenreich20243sq propose an iterative, semi-automatic process leveraging Large Language Models (LLMs) to generate and evaluate multiple software architecture candidates directly from textual requirements. This approach is pivotal for compliance-aware design, as architectural decisions often embed fundamental compliance principles, allowing for proactive adherence to security or data privacy guidelines from the blueprint stage. However, the effectiveness of such LLM-driven architectural generation in fully capturing and enforcing complex, non-functional compliance requirements, which are often ambiguous and context-dependent, remains a significant challenge, requiring extensive prompt engineering and human refinement **?**. The inherent "black-box" nature of LLMs also complicates the auditability and explainability of why a particular architectural decision was made, a critical aspect for regulatory compliance **?**.

The application of GenAI extends significantly into code generation, moving beyond mere assistance to autonomous creation. A comprehensive review of AI techniques for Automated Code Generation (ACG) highlights advancements in Machine Learning (ML), Natural Language Processing (NLP), Deep Learning (DL), and Evolutionary Algorithms (EAs) in translating requirements into executable code **?**. While these techniques demonstrate potential for efficiency, ensuring the correctness, quality, and, crucially, the compliance of generated code remains a significant challenge. General GenAI coding tools, such as GitHub Copilot and ChatGPT, despite offering productivity gains, frequently generate code with security vulnerabilities, logical flaws, and "hallucinations" (incorrect or nonsensical outputs) **???**. This necessitates thorough developer review and testing, as AI-generated code often requires significant correction to meet quality and security standards **?**. Empirical studies have even shown that developers using AI assistants can produce *less secure code* and exhibit increased overconfidence in its security, highlighting a critical human factor in the compliance chain **???**. This human-AI interaction dynamic poses a direct threat to the 'compliance-by-design' paradigm if not properly managed, as the ultimate responsibility for compliant code still rests with the human developer.

A significant advancement in embedding proactive compliance comes from frameworks where LLMs can autonomously create and verify their own programmatic tools. sauvola2024zw7 introduce the LLMs AsToolMakers (LATM) framework, a closed-loop system where LLMs generate, verify (through unit tests and self-correction), and manage reusable Python-based tools. This innovative architecture, which separates a powerful "tool maker" LLM from a lightweight "tool user" LLM, directly addresses the limitations of high inference costs for repetitive tasks and the lack of reusability in LLM-generated code. For compliance-aware design and code generation, LATM represents a transformative capability: an LLM could autonomously generate a specific programmatic tool to check for adherence to a particular security policy, a data minimization principle, or an architectural guideline. This self-generated and verified tool could then be applied to generated designs or code, or even provided to developers, ensuring that compliance checks are not only automated but also dynamically created and validated by the AI itself, fostering true 'compliance-by-design'. However, the robustness of LATM's self-correction mechanism against complex, ambiguous, or evolving compliance rules, which often lack clear, deterministic test cases, needs further rigorous validation. The framework's reliance on unit tests for verification might also fall short for non-functional compliance properties that require broader system-level analysis.

To address the inherent limitations of probabilistic GenAI outputs for compliance-critical systems, researchers are exploring integrations with more deterministic approaches. The concept of "correct-by-construction" from formal methods and program synthesis offers a contrast to GenAI's heuristic nature. While GenAI excels at generating diverse solutions, formal methods aim for provably correct outputs based on specifications. fernandez2024lee advocate for teaching software verification alongside GenAI, emphasizing the critical need for developers to understand and verify AI-generated code for security and correctness. This suggests that even with "compliance-aware" generation, a robust verification step is indispensable. Furthermore, Model-Driven Engineering (MDE) and Domain-Specific Languages (DSLs) provide structured ways to embed compliance rules directly into models, which can then be used to generate code with stronger guarantees ?. Integrating GenAI with MDE, where LLMs generate initial models or transformations that are then formally verified or processed by MDE tools, could offer a hybrid approach to achieve more reliably compliant designs and code. This would leverage GenAI's creativity for initial design exploration while employing MDE's rigor for rule enforcement and traceability.

In conclusion, Generative AI is rapidly evolving from a tool for general software development assistance to a powerful enabler of compliance-aware design and code generation. The ability of LLMs to semi-automatically generate software architectures from requirements ? and, more profoundly, to autonomously create and verify their own programmatic tools for specific compliance checks ?, marks a significant step towards embedding compliance from the outset. However, the probabilistic nature of GenAI, coupled with challenges such as hallucinations, security vulnerabilities, and human overconfidence, necessitates a critical approach. Future advancements must focus on enhancing the verifiability and explainability of AI-generated artifacts, integrating GenAI with formal methods and model-driven approaches for stronger compliance guarantees, and designing AI assistants that actively mitigate human-induced security risks. The trajectory towards 'compliance-by-design' through autonomous, verifiable, and proactive AI-driven solutions is clear, promising more secure and compliant software systems, but requires continuous innovation in evaluation, integration, and human-AI collaboration.

5.3 Integrating AI-Driven Compliance into DevOps and Agile Workflows

The dynamic and iterative nature of modern software development, characterized by DevOps and Agile methodologies, presents a significant challenge for maintaining continuous regulatory and quality compliance. This necessitates a paradigm shift from reactive, end-of-cycle compliance checks to a proactive, continuous, and integrated approach. AI-driven solutions are emerging as pivotal enablers for embedding compliance mechanisms seamlessly throughout the software development lifecycle, facilitating early detection of non-compliance and providing rapid feedback without impeding agility. This integration transforms compliance into an intrinsic, rather than an extrinsic, part of the development process.

Initial advancements in this domain have focused on leveraging AI to automate and enhance compliance activities within Continuous Integration/Continuous Delivery (CI/CD) pipelines. AI-driven CI/CD systems automate the integration, testing, packaging, and deployment processes, significantly reducing manual errors and accelerating software delivery cycles ?. Within these pipelines, AI-powered tools are integrated as automated gates to perform continuous security compliance checks, identify vulnerabilities in code and configurations, and detect deviations from established quality standards in real-time ?. This "shift-left" approach to compliance, where checks are performed as early as possible, allows development teams to address issues immediately, preventing costly rework later in the cycle. For instance, AI-driven solutions can analyze Infrastructure-as-Code (IaC) configurations against predefined security policies or regulatory standards, flagging non-compliant patterns before deployment within the CI/CD pipeline ?. The integration of AI into these automated gates ensures that compliance is not an afterthought but a continuous, verifiable validation step.

Beyond the technical automation of CI/CD, AI-driven compliance mechanisms are increasingly being integrated into the human-centric aspects of Agile workflows, enabling a more proactive, "compliance-by-design" approach. Large Language Models (LLMs) and Generative AI are instrumental in bridging the gap between complex, natural language regulatory texts and actionable software requirements or formalizable policies ??. In Agile contexts, this translates to AI-assisted activities during sprint planning and backlog refinement. For example, AI assistants can help product owners and development teams systematically derive non-functional requirements (NFRs) related to trustworthiness, ethics, privacy, safety, and security from regulatory documents, integrating these compliance considerations directly into user stories and acceptance criteria ?. The concept of "Responsible AI" (RAI) is operationalized through software engineering patterns and integrated into MLOps (Machine Learning Operations) workflows, which are the DevOps equivalent for AI systems. This ensures that ethical and compliance principles are considered at every stage of the AI system's lifecycle, from data engineering to continuous deployment and monitoring, thereby embedding compliance into the very fabric of AI development ??. Furthermore, multi-agent AI systems are being explored to automate entire segments of the Software Development Lifecycle (SDLC), including requirements engineering (generating and prioritizing user stories) and architectural design, with integrated compliance agents that ensure adherence to standards and regulations throughout the process ?. This represents a significant step towards embedding compliance deeply within the iterative design and development phases of Agile.

The integration of AI also extends to enhancing specific compliance tasks within the development workflow, often through human-AI collaboration. LLMs can be utilized for security-related tasks, such as vulnerability detection during code reviews or as pre-commit hooks, providing real-time assistance to developers. While current LLM outputs for vulnerability detection may sometimes be generic and require human oversight, their potential for augmenting developers' capabilities in maintaining security compliance is significant ??. Similarly, AI-driven testing automation, including the generation of diverse test cases and simulation of realistic user behaviors, indirectly supports compliance by identifying faults related to accessibility, data privacy, or functional correctness that might otherwise lead to non-compliance with regulatory standards ?. The success of integrating these AI tools into fast-paced development environments heavily relies on their compatibility with existing workflows, which has been identified as a predominant driver for Generative AI adoption among software engineers ?.

Despite the substantial benefits, integrating AI-driven compliance into fast-paced DevOps and Agile environments presents unique challenges. The scalability and generalizability of specific AI models across diverse and evolving compliance domains remain a significant concern. Regulations are often nuanced, context-dependent, and subject to frequent changes, requiring AI models to be highly adaptive and capable of learning new rules rapidly, a challenge highlighted in the need for revised AI lifecycle models in regulated sectors ?. The computational overhead of advanced AI techniques, especially for continuous, real-time analysis across large codebases and complex systems, can be substantial. Moreover, effectively integrating diverse AI tools into existing, complex DevOps toolchains and Agile processes requires careful architectural planning, robust interoperability, and often, significant organizational change management ?. A critical aspect for successful integration, particularly in highly regulated sectors, is ensuring that the AI-driven compliance decisions are transparent and auditable. While detailed explainability is covered in Section 6.1, the need for AI systems to provide clear, legally sufficient justifications for their compliance assessments is paramount for developer trust, regulatory acceptance, and effective debugging within the integrated workflow. Furthermore, human factors, such as user overconfidence when using AI assistants, can inadvertently lead to the introduction of less secure code, necessitating usable security research and developer education to mitigate such risks within AI-assisted workflows **?**.

In conclusion, the integration of AI-driven compliance into DevOps and Agile workflows is rapidly evolving, moving from reactive detection within CI/CD pipelines to proactive, compliance-aware design and continuous, transparent verification. This shift is crucial for maintaining compliance in fast-paced, iterative development environments by embedding compliance as an intrinsic part of the development process. While AI offers substantial benefits in terms of early detection, automated enforcement, and accelerated development, challenges persist regarding the scalability and generalizability of specific AI models, the computational overhead of advanced techniques, the critical need for robust workflow compatibility, and legally sufficient explainability within the integrated workflow. Future research must focus on developing adaptive AI models that can rapidly learn and respond to evolving regulations, enhancing the trustworthiness and auditability of AI decisions, and establishing ethical AI governance frameworks to ensure responsible and effective compliance automation seamlessly integrated into the development lifecycle.

6 Critical Considerations for Trustworthy AI-Driven Compliance Systems

6.1 Explainable AI (XAI) for Compliance Auditing and Trust

The increasing integration of Artificial Intelligence (AI) into critical software development processes, particularly in highly regulated domains, necessitates a robust solution to the inherent "black box" problem of many AI systems. For AI-driven compliance assessments to achieve widespread adoption and legal sufficiency, they must be not only accurate but also transparent, interpretable, and trustworthy. This imperative drives the application of Explainable AI (XAI) techniques, which aim to provide human-understandable reasons for AI-driven decisions, crucial for regulatory auditing, debugging AI models, and building trust among stakeholders. This subsection explores the role of XAI in achieving these goals, drawing on recent literature that operationalizes trustworthy AI and addresses the practical challenges of integrating ethical principles into the software development lifecycle.

The concept of explainability is a cornerstone of broader frameworks for Trustworthy AI (TAI) and Responsible AI (RAI), moving beyond abstract ethical principles to concrete implementation. **?** introduces POLARIS, a holistic framework that explicitly includes Explainability as one of its four foundational pillars (alongside Privacy, Security, and Fairness). This framework is designed to guide AI practitioners throughout the entire Software Development Life Cycle (SDLC), addressing the critical gap where most existing ethical AI guidelines offer high-level principles but lack actionable strategies or tools for technical implementation. Similarly, **?** conducted an empirical study with AI scientists and engineers, identifying "Transparency & Explainability" as one of the most frequently discussed AI ethics principles, underscoring its practical importance. Their work proposes operationalized patterns to integrate responsible AI considerations, including explainability, across the AI system lifecycle, from requirements engineering to deployment. **?** further supports this, highlighting transparency and explainability as key ethical aspects in their empirical investigation of implementing responsible AI.

The need for XAI is particularly acute in compliance auditing, where decisions must be legally defensible and auditable. **?** emphasizes the importance of moving beyond abstract ethical principles to "verifiable claims" about AI systems. They propose a "toolbox" of mechanisms, including software-based solutions like audit trails and interpretability methods, specifically focused on supporting risk assessment and auditing. For AI systems making compliance decisions, interpretability is not merely about understanding the model; it's about providing evidence that the decision aligns with regulatory mandates, organizational policies, and ethical standards. This directly addresses the challenge posed by legal ambiguities and liability concerns for AI errors, as highlighted by **?**, where the lack of clear explanations can hinder accountability and legal recourse.

XAI techniques contribute to building trust by making AI decisions transparent, thereby mitigating concerns about algorithmic bias and unfair outcomes. ? notes that automated decision-making algorithms have displayed evidence of bias, lack ethical governance, and limit transparency, leading to unfair outcomes. By providing insights into how an AI model arrived at a particular compliance assessment, XAI can help identify and rectify such biases, ensuring fairness and non-discrimination. ? reinforces this by proposing Non-Functional Requirements (NFRs) for socially responsible software, where Transparency is identified as a crucial enabler for Trust and a mitigant for Legal Disputes. In the context of compliance, this means explanations can demonstrate that an AI system adheres to principles like fairness and accountability, which are often implicit or explicit requirements in regulatory frameworks such as GDPR ?.

Despite the clear necessity, the practical implementation of XAI for compliance faces significant challenges. A rapid review of Responsible AI frameworks by ? revealed a critical gap: only a small percentage of frameworks offer practical tools to support RAI implementation, and there is a stark imbalance in SDLC coverage, with most focusing on early requirements elicitation while neglecting later stages like testing and deployment. This indicates a broader deficiency in actionable guidance for integrating explainability throughout the entire software development lifecycle. Furthermore, while XAI methods like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) can provide local explanations for individual predictions, the question of "explanation sufficiency" for legal and regulatory purposes remains complex. What constitutes an adequate explanation for an auditor or a legal team may differ significantly from what a developer needs for debugging, requiring context-aware and audience-specific explanations. The trade-off between model fidelity and interpretability, the scalability of XAI methods to complex, large-scale AI systems, and the potential for explanations themselves to be misleading or incomplete are ongoing research areas. ? further highlights a notable gap between high-level AI ethics guidelines and actual industrial practice, suggesting that operationalizing XAI effectively requires overcoming significant organizational and technical hurdles.

In conclusion, XAI is not merely a desirable feature but a critical enabler for the trustworthy and legally sufficient adoption of AI in compliance-critical software development. While frameworks like POLARIS **?** and empirical studies **??** underscore the foundational role of explainability in responsible AI, significant work remains to translate these principles into scalable, actionable, and legally robust XAI techniques. Future research must focus on developing context-aware XAI methods that cater to the diverse needs of stakeholders (developers, auditors, legal teams), providing explanations that are not only technically sound but also legally sufficient and easily auditable. This includes integrating XAI seamlessly into every phase of the SDLC, from design to deployment, to ensure continuous compliance assurance and foster genuine trust in AI-driven compliance systems.

6.2 Blockchain for Immutable Compliance Evidence and Traceability

The increasing reliance on Artificial Intelligence (AI) for automating compliance processes, spanning from proactive monitoring to complex policy interpretation **???**, necessitates robust mechanisms to ensure the trustworthiness and auditability of these systems. While AI significantly enhances the efficiency and scope of compliance checking, moving beyond foundational automated methods **?**, it also introduces challenges related to the "black box" nature of AI decision-making and the potential for data manipulation. This creates a critical demand for verifiable records and an unalterable chain of custody for compliance artifacts, particularly in highly regulated environments where demonstrable adherence to standards is paramount. Blockchain technology emerges as a powerful solution to address these concerns by providing an immutable and transparent ledger for all compliance-related activities and evidence generated by AI systems, thereby contributing to the broader goal of Trustworthy AI (TAI) **?**.

Blockchain's distributed, cryptographic, and immutable properties are uniquely suited to enhance the integrity and auditability of AI-driven compliance. It offers a secure platform to record every step of the compliance process, from the ingestion of regulatory rules and the configuration of AI models to the generation of compliance evidence and the final decision outcomes. **?** proposed a seminal blockchain-based framework specifically targeting the immutability and traceability of compliance evidence generated by AI systems in software development. This framework leverages blockchain to create an unalterable record of all relevant data, including AI model inputs, outputs, intermediate decisions, and the compliance artifacts produced, ensuring these records cannot be tampered with post-generation.

Extending this, **?** comprehensively surveys how blockchain can make AI trustworthy across its software development lifecycle (SDLC), classifying its contributions into planning, data collection, model development, and system deployment/use. For instance, in the **data collection** stage, blockchain can ensure data transparency, privacy, and accountability by immutably recording data provenance, consent mechanisms, and transformations applied to training datasets. This is crucial for verifying that AI models are trained on compliant, ethically sourced data. During **model development**, blockchain can log AI model versions, configuration parameters, training logs, and validation results, thereby enhancing model transparency, robustness, and fairness **?**. This addresses the critical need for robust documentation and risk assessment in AI lifecycle models, as highlighted by **?**, which identified documentation and model risk assessment as frequently overlooked yet essential stages in regulated environments like fintech. By recording these artifacts on a blockchain, organizations can create an unalterable audit trail for model governance.

The integration of blockchain establishes a verifiable and transparent audit trail, which is paramount for regulatory scrutiny and building confidence in automated compliance systems. By timestamping and cryptographically linking each piece of evidence to the blockchain, it becomes possible to trace the provenance of every compliance assertion back to its origin, ensuring an unalterable chain of custody. Concrete examples of compliance evidence that can be recorded on-chain include:

- **Hashed Code Commits and Software Artifacts**: Cryptographic hashes of source code, configuration files, and deployment scripts, ensuring their integrity.
- **AI Model Metadata**: Version identifiers, training data hashes, hyperparameter

settings, and performance metrics of AI models used for compliance checks.

- **Compliance Reports and Decisions**: AI-generated compliance reports, static analysis results, security vulnerability scans, and records of human overrides or approvals.

- **Regulatory Interpretations**: The specific regulatory rules or policies (e.g., GDPR articles) that an AI system was configured to enforce, potentially linked to their natural language interpretations.

This not only strengthens accountability for AI-driven compliance decisions but also provides irrefutable proof of adherence to regulations. For instance, if an AI system flags a potential non-compliance or certifies compliance, the underlying data, the AI's reasoning (if logged), and the final determination can all be immutably recorded, offering unparalleled transparency to auditors and stakeholders.

To achieve greater technical depth, the application of **smart contracts** is crucial. Smart contracts can be used to encode compliance rules and automate their execution on the blockchain. For example, a smart contract could automatically verify if a hashed code commit meets predefined security standards or if a dataset's provenance adheres to privacy regulations, recording the outcome immutably. This moves beyond mere data storage to active, automated compliance enforcement. Furthermore, the choice of **blockchain architecture** is critical for enterprise compliance. While public blockchains offer maximum decentralization, permissioned or consortium blockchains (e.g., Hyperledger Fabric) are often preferred in regulated environments due to their controlled access, enhanced privacy features, and higher transaction throughput. These architectures allow organizations to maintain data confidentiality while still leveraging blockchain's immutability for auditability.

While blockchain primarily addresses the integrity and traceability of compliance evidence, it complements other approaches aimed at enhancing trust in AI. For example, Explainable AI (XAI) techniques, as explored in the preceding subsection, focus on providing transparent and understandable reasons for AI's compliance decisions, thereby tackling the "black box" problem from an interpretability perspective. Together, blockchain and XAI offer a comprehensive strategy: blockchain ensures *what* happened is immutably recorded and traceable, while XAI clarifies *why* it happened. This combined approach directly supports the principles of transparency and accountability identified as critical yet often lacking in Responsible AI frameworks **?**.

Despite its significant advantages, the practical implementation of blockchain for AI-driven compliance evidence faces several challenges. These include the computational overhead associated with maintaining a distributed ledger, scalability issues when dealing with the vast amounts of data generated by AI systems, and the need for standardized protocols for recording diverse compliance artifacts on-chain. Furthermore, legal and regulatory acceptance of blockchain-based evidence is still evolving, requiring clear guidelines and frameworks. Future research must focus on optimizing blockchain scalability, developing interoperable standards for diverse compliance artifacts, and exploring hybrid architectures that balance on-chain immutability with off-chain data storage for sensitive information, thereby fully realizing its potential in securing AI-driven compliance.

6.3 Human Factors and Risks in AI-Assisted Secure Development and Auditability

The integration of Artificial Intelligence (AI) into software development fundamentally alters developer workflows, introducing a complex interplay of human factors that critically impact the security, compliance, and, crucially, the auditability of the entire development process. While Section 4.2 addresses the direct introduction of vulnerabilities by AI systems and developer overconfidence in code security, this subsection delves deeper into how human-AI interaction patterns, trust dynamics, and scrutiny practices influence the *trustworthiness* and *verifiability* of AI-assisted development artifacts and processes. It underscores the imperative for usable security research, developer education, and AI assistant designs that actively mitigate human-induced vulnerabilities, ensuring that AI tools enhance, rather than compromise, overall software security and compliance, and that the resulting artifacts remain auditable.

Empirical studies reveal a nuanced landscape of human-AI collaboration. Developers often exhibit a general mistrust in the security of AI suggestions due to overall quality concerns, yet paradoxically, AI assistants are widely used for security-critical tasks like code generation, threat modeling, and vulnerability detection **?**. This highlights a significant gap between perceived risk and actual usage, where the convenience and productivity gains often outweigh explicit security concerns in practice. For instance, large-scale telemetry analysis of GitHub Copilot users demonstrates substantial productivity increases, particularly for less experienced developers **?**. While beneficial for efficiency, this finding poses a heightened security risk: less experienced developers, who may possess less inherent security expertise, are leveraging tools that can generate insecure code, increasing the likelihood of vulnerabilities being introduced. This dynamic complicates auditability, as the human review layer, intended as a safeguard, may be less effective due to a lack of expertise or over-reliance on AI.

The challenge is further exacerbated by the cognitive biases and trust dynamics inherent in human-AI interaction. **?** demonstrated that developers using AI assistants wrote significantly less secure code and, critically, exhibited increased overconfidence in their code's security. This overconfidence directly impacts auditability: if developers *believe* their AI-assisted code is secure, they may not apply the rigorous scrutiny or document the comprehensive review steps necessary for a robust audit trail. **?** further explored trust dynamics, finding that developers primarily evaluate AI code suggestions based on comprehensibility and perceived correctness, but identified a significant lack of real-time support for robust trust evaluation. This results in developers frequently altering AI suggestions, accepting only about 52

These human factors present profound challenges for establishing trustworthy AI-driven compliance systems and ensuring auditability. The "black box" nature of many AI models, combined with human overconfidence and inconsistent scrutiny, can lead to a lack of transparency regarding *why* certain code was generated or accepted, *who* is accountable for its security, and *how* compliance was verified. This necessitates a shift towards designing AI-assisted development processes that inherently support verifiable claims and robust audit trails.

To address these issues, several approaches are proposed to enhance trustworthiness and auditability. **?** advocate for mechanisms supporting verifiable claims in AI development, emphasizing the need for comprehensive audit trails, interpretability methods tailored for risk assessment and auditing, and secure hardware for machine learning to enhance accountability and transparency. Such audit trails must capture not only the final code but also the AI prompts, AI responses, human modifications, and the rationale behind acceptance or rejection, creating an immutable record of the human-AI collaboration. Building on this, **?** propose operationalized patterns for Responsible AI, integrating continuous monitoring and system-level ethical considerations throughout the AI system lifecycle. This process-oriented view is crucial for embedding auditability from inception, rather than as an afterthought. Furthermore, **?** introduced a maturity model (AI-MM) for trustworthy AI software development, providing practical guidelines for enhancing maturity levels in AI projects, which implicitly leads to better-defined processes and improved auditability.

Beyond process-level changes, technical solutions within AI tools themselves are vital. **?** envisions future AI-driven software engineering where Explainable AI (XAI) provides justifications for design suggestions, aiding human decision-making and building trust. They also highlight Metamorphic Testing as a critical technique for verifying AI-generated code, especially for addressing the oracle problem, thereby enhancing the verifiability of AI outputs. The concept of "security-aware" AI assistants, actively designed to mitigate vulnerabilities and user overconfidence, as implied by **?**, is paramount. This could involve AI flagging potential security risks in its own suggestions or prompting developers for explicit security review.

Finally, developer education remains a critical component, though its efficacy against deep-seated cognitive biases requires careful consideration. **?** advocate for integrating basic AI knowledge and traditional software verification steps into early computer science curricula, equipping future developers with the skills to critically evaluate AI-generated code. However, education alone may not be sufficient to counteract automation bias or overconfidence; technical guardrails and robust process frameworks are equally essential to ensure that the human element in AI-assisted development consistently contributes to, rather than detracts from, software security and auditability.

In conclusion, the human element in AI-assisted secure development presents a double-edged sword: while human oversight is indispensable for validating AI outputs, inherent biases like overconfidence, coupled with a lack of robust support for trust evaluation and audit trails, can undermine the trustworthiness and auditability of the entire process. Addressing this requires a multi-faceted approach, combining transparent AI designs, comprehensive developer education, and structured frameworks for verifiable claims and continuous monitoring throughout the software development lifecycle.

7 Conclusion and Future Directions

7.1 Synthesis of Key Findings

The extensive literature on AI for software development compliance reveals a dynamic field characterized by a compelling dual trajectory: the relentless advancement of AI capabilities to augment and automate software engineering tasks, and the simultaneous, urgent imperative to ensure these AI systems are developed and deployed responsibly, ethically, and compliantly. This review has traced the intellectual progression from foundational AI capabilities and initial productivity gains to the identification of critical risks, the development of conceptual frameworks for responsible AI, and the emergence of advanced AI-driven detection and proactive design solutions. The field is rapidly maturing, demonstrating the interconnectedness of its various research streams and underscoring the comprehensive scope of AI's impact on the entire software development lifecycle.

Initially, research focused on establishing the bedrock for "AI for Code," addressing data scarcity through large-scale datasets and demonstrating AI's capacity to significantly enhance developer productivity. Early empirical studies consistently showed that AI assistants could accelerate task completion and democratize access to advanced coding practices, with less experienced developers often benefiting disproportionately. This initial optimism, however, quickly gave way to a more nuanced understanding of human-AI interaction, revealing that while AI boosts individual productivity, it also reshapes team dynamics, information-seeking behaviors, and introduces user experience challenges related to validating AI-generated content ????. Concurrently, the technical frontier pushed towards increasingly autonomous AI agents and platforms capable of complex software engineering tasks, including self-tooling and multimodal reasoning.

Crucially, as AI integration deepened, the focus broadened from mere efficiency to the profound trustworthiness and compliance challenges inherent in AI-assisted development. A pivotal finding, empirically validated by multiple studies, highlighted a significant tension: developers, when assisted by AI, often produce less secure code and exhibit increased overconfidence in its security, despite generally mistrusting AI suggestions for critical tasks ??. This exposed a severe, counterintuitive compliance risk. Beyond security, the environmental footprint of AI systems emerged as a critical concern, with research providing holistic frameworks for quantifying carbon emissions across the AI lifecycle and advocating for "green AI" architectures ??. Furthermore, empirical investigations consistently revealed a notable gap between abstract AI ethics guidelines and practical industry implementation, particularly concerning societal well-being, diversity, non-discrimination, and fairness ????. These findings underscored that the "compliance" aspect extends not only to the software being built but also to the inherent qualities and development processes of the AI systems themselves.

In response to these identified problems, the field has progressed towards developing more robust conceptual frameworks and technical solutions for proactive design and governance. Visionary paradigms, such as "AI-native Software Engineering," articulate a future where AI acts as an intelligent, goal-driven teammate, emphasizing intent-first development and knowledge-driven models to ensure consistency and quality across the SDLC ???. However, a critical gap remains between these aspirational visions and the current reality of practical, validated solutions, which often address point problems rather than holistic, end-to-end compliance. To bridge this, research has proposed concrete frameworks for moving beyond abstract AI ethics to verifiable claims, supported by mechanisms like audit trails and interpretability ?. This shift is echoed in the call for revised AI lifecycle models that explicitly incorporate critical stages like data collection, feasibility studies, robust documentation, continuous model monitoring, and comprehensive risk assessment, particularly for heavily regulated domains like fintech ?. Such models emphasize the need for legal mandates and external oversight to ensure compliance with human rights standards and broader societal values ??.

The application of AI for automated compliance detection has evolved from traditional AI/ML techniques for code and artifact analysis to leveraging Large Language Models (LLMs) for complex regulatory interpretation and policy-to-code mapping. Generative AI is increasingly explored for compliance-aware design, aiming to embed compliance proactively into software components rather than merely detecting violations post-factum. The integration of these AI-driven mechanisms into DevOps and Agile workflows signifies a move towards continuous compliance throughout the entire development lifecycle. Yet, the trustworthiness of these AI-driven compliance systems themselves remains paramount. Explainable AI (XAI) techniques are critical for providing human-understandable justifications for AI decisions, essential for auditing and building trust. Similarly, blockchain technology offers a promising avenue for immutable compliance evidence and traceability, bolstering accountability in regulated environments ?. Furthermore, addressing human factors, such as developer overconfidence and the need for rigorous scrutiny of AI-generated code, is crucial for mitigating risks and ensuring the auditability of AI-assisted secure development ?.

In conclusion, the journey of AI for software development compliance reflects a rapid maturation, transitioning from initial explorations of AI's utility to a sophisticated engagement with its profound implications. The field has successfully identified the dual nature of AI's impact, moving from problem identification—such as security vulnerabilities, ethical gaps, and environmental concerns—to the development of conceptual frameworks, technical solutions, and empirical validations. However, a persistent tension exists between the pursuit of efficiency and the imperative for transparency, auditability, and human interpretability. While significant progress has been made in automating detection and envisioning proactive design, the overarching challenge lies in seamlessly integrating these advancements to create inherently compliant and trustworthy AI-driven SDLCs that not only leverage AI's technical capabilities but also robustly address ethical, regulatory, and human demands. This necessitates continuous interdisciplinary research, robust validation in real-world settings, and a commitment to responsible innovation to navigate the complexities of this evolving landscape.

7.2 Unresolved Tensions and Theoretical Gaps

The rapid advancement of AI in software engineering, particularly in compliance-related tasks, has unveiled significant unresolved tensions and theoretical gaps that demand further inquiry. While AI models offer unprecedented efficiency and power, their application in regulated environments often clashes with fundamental imperatives for transparency, auditability, and human interpretability.

A primary tension lies in the inherent trade-offs between the efficiency and power of complex AI models and the critical need for transparency and auditability. Advanced AI agents, such as those capable of autonomously creating their own tools **?** or operating as generalist software developers in sandboxed environments **?**, demonstrate immense potential for productivity gains **?**. However, this sophistication often comes at the cost of explainability. The "black box" nature of many powerful AI models, particularly Large Language Models (LLMs) used for tasks like regulatory interpretation **?** or proactive compliance design **?**, introduces challenges in providing clear, human-understandable justifications for their decisions. This directly conflicts with the need for verifiable claims in trustworthy AI development **?**, where external scrutiny and accountability are paramount. Empirical evidence further exacerbates this tension, as studies show that AI assistants can lead to significantly less secure code and increased user overconfidence **?**, directly undermining the auditability and safety required in regulated software development. While Explainable AI (XAI) techniques are proposed for compliance auditing **?**, their practical integration into complex, real-time systems and ensuring the legal sufficiency of explanations remain significant hurdles.

Another critical gap concerns the scalability and generalizability of specific AI models across diverse compliance domains. While foundational platforms like the Arcade Learning Environment **?** and large datasets like CodeNet **?** aim for general AI capabilities, their focus often remains on functional performance rather than comprehensive compliance implications. Automated compliance detection and enforcement systems **???** aim to reduce manual effort, yet they frequently struggle with high false positive rates and adapting to new or nuanced regulatory contexts without extensive retraining. For instance, while environmental compliance tools measure the carbon intensity of AI workloads **??**, their methodologies are highly specific and do not readily generalize to other compliance areas like data privacy or security. The introduction of benchmarks like SWE-bench Multimodal **?** highlights the current limitations of AI systems in generalizing across visual and multi-language software domains, suggesting that generalizability for compliance, which often involves complex, domain-specific rules, is an even more formidable challenge.

The dynamic nature of regulatory environments presents a significant challenge for AI-driven compliance solutions. Regulations evolve rapidly, often outpacing the development and validation cycles of complex AI models. While LLMs show promise in interpreting regulatory texts **?**, their potential for "hallucinations" and the computational cost of continuous adaptation make reliable, real-time responses to evolving legal frameworks difficult without substantial human oversight. The existing literature on automated compliance often overlooks the mechanisms for rapid model adaptation and validation in response to regulatory shifts, leaving a critical gap in ensuring continuous compliance.

Finally, there is a pronounced need for more robust, long-term empirical validation of proposed frameworks and solutions in real-world industrial settings. Many frameworks, such as POLARIS for trustworthy AI development **?**, offer actionable guidelines across the Software Development Life Cycle (SDLC) but are currently supported by only "initial validation." A systematic review of Responsible AI frameworks empirically confirms a severe lack of actionable tools and comprehensive SDLC coverage, particularly in later phases like design, development, testing, and deployment **?**. This indicates that while conceptual solutions are emerging, their practical effectiveness and long-term impact in diverse, complex industrial environments remain largely unproven. Even controlled experiments demonstrating AI's productivity benefits **?** or security risks **?** often operate within specific task contexts or participant pools, limiting their generalizability to the full spectrum of real-world industrial compliance scenarios.

These unresolved tensions and theoretical gaps collectively represent fertile ground for future theoretical and empirical inquiry. Addressing them demands innovative solutions that can effectively balance the power of AI with the non-negotiable requirements of transparency, auditability, and human interpretability, while also ensuring scalability, adaptability to evolving regulations, and rigorous, long-term validation in complex industrial settings.

7.3 Practical Challenges and Ethical Implications

The increasing reliance on Artificial Intelligence (AI) for software development compliance, while offering transformative potential, simultaneously introduces a complex array of practical challenges and profound ethical implications. These issues necessitate continuous vigilance and the development of robust, responsible deployment strategies to navigate the intricate landscape of AI-driven software engineering.

A primary practical challenge lies in the overhead associated with integrating AI tools into existing development pipelines and ensuring the reliability of their outputs. While studies like **?** demonstrate substantial productivity gains from AI assistants, the integration is rarely seamless. "Workflow compatibility" is a predominant driver for generative AI adoption, implying significant integration hurdles if tools disrupt established practices **?**. This is further elaborated by **?**, which developed a "Theory of AI Tool Adoption" identifying numerous individual and organizational challenges, including the effort required to adapt workflows and the need for organizational support. Developers frequently face an "overhead in prompt engineering for quality output" and a persistent "need for double-checking GenAI-generated content" **?**, which can partially offset efficiency gains. **?** highlights the "burden of validating AI-generated information" and notes that AI tools can exhibit "adaptive/inconsistent responses that erode trust," requiring developers to cross-reference multiple AI systems or traditional sources. Quantifying this overhead, **?** introduced a "human intervention level" metric, showing that even advanced LLMs still require human oversight and modifications to integrate their outputs effectively. Furthermore, AI tools often struggle with "complex multi-file tasks" and "large proprietary contexts," limiting their utility in intricate enterprise projects **?**. The systematic review by **?** on Human-AI Experience in IDEs corroborates these findings, noting "increased time spent on result verification" and concerns about over-reliance, underscoring the integration and validation burden. **?** further identifies challenges such as "no matching use cases" and "unforeseen benefits" that slow down GenAI adoption, particularly in contexts where specific, tailored solutions are required.

Beyond integration and validation, the computational cost of advanced AI models presents a significant practical barrier. As discussed in Section 3.3, the current generation of Foundation Models (FMs) relies on vast amounts of unstructured data, leading to "high computational costs, energy consumption, [and] data redundancy" **?**. This inefficiency not only impacts the economic viability of widespread deployment but also raises environmental concerns, necessitating the development of "green AI" architectures. Future visions for "AI-native Software Engineering" (SE 3.0) acknowledge this, calling for "cheaper and smarter code models" and "knowledge-driven models" to overcome the data-driven inefficiencies of current FMs **?**.

A critical, often underestimated, practical challenge is the inherent complexity of encoding comprehensive compliance rules into a machine-interpretable format. While Section 5.1 explored leveraging LLMs for regulatory interpretation, the actual translation of ambiguous, natural language legal texts into precise, executable rules remains formidable. **?** highlights that even with generative AI, applying principled software engineering techniques is necessary to enhance AI-driven deductive legal reasoning of complex statutes, treating LLMs as interpreters of natural-language programs. This implies that the 'program' (the compliance rules) itself needs careful engineering and formalization, which is a non-trivial task given the nuances, exceptions, and evolving nature common in legal and regulatory frameworks. The ambiguity and dynamic nature of regulations mean that a static encoding is often insufficient, requiring continuous adaptation and expert intervention to maintain accuracy and legal sufficiency.

Ethically, the deployment of AI in software development compliance raises profound concerns, starting with the potential for 'hallucinations' and inherent bias in generative AI. Generative AI models are prone to producing incorrect or nonsensical information, termed "hallucinations," and their performance is heavily reliant on the quality and representativeness of their training data, which can lead to "algorithmic bias" **?**. This "black box problem," as discussed in Section 6.1, limits interpretability and trustworthiness, especially when AI is used for critical compliance tasks. The concept of "vibe coding," where AI generates large portions of code, introduces risks of "black box codebases" and "ecosystem bias" if the underlying models are not transparent and fair **?**. **?** also underscores concerns regarding the quality, security, and privacy of AI-generated code, emphasizing the need for robust verification methods like metamorphic testing.

A critical ethical concern is the security of AI-generated code and the accountability for errors. As elaborated in Section 6.3, empirical studies have shown that AI assistants can lead developers to write "significantly less secure code" and foster "increased user overconfidence" regarding security flaws **?**. This finding is echoed by **?**, whose qualitative study revealed a paradox: developers generally mistrust the security of AI suggestions but widely use them for security-critical tasks, necessitating rigorous manual review. **?** warns that "novice programmers leveraging AI-code-generation without proper understanding of syntax or logic can create 'black box' code with significant security vulnerabilities." This creates "responsibility gaps" where accountability for AI-generated errors becomes ambiguous **?**. The challenge of establishing "trust dynamics" in AI-assisted development, particularly in evaluating the trustworthiness of AI suggestions, is explored by **?**, highlighting the need for better support in real-time trust decisions. **?** emphasizes that conventional engineering methods are inadequate for guaranteeing dependability (safety, security, privacy) in autonomous systems, a challenge exacerbated by the integration of AI.

Furthermore, data privacy and intellectual property are significant ethical battlegrounds. The use of vast datasets for training AI models raises questions about the privacy of the data subjects and the potential for unintended data leakage. Similarly, the generation of code by AI systems complicates intellectual property rights, with "unresolved legal questions concerning ownership and application of copyright, patent, and trademark laws to AI-generated content" **?**. The broader legal and ethical implications of human-AI collaboration in programming are also a growing concern **?**. Effective AI governance frameworks are crucial to address these, requiring algorithmic auditing and adherence to international principles **?**. **?** advocates for a human rights-centered design, deliberation, and oversight approach, integrating human rights norms at every stage of AI system design and deployment, adapting technical methods for verification and auditing to ensure compliance with these fundamental values.

Alarmingly, empirical studies reveal a significant disconnect between theoretical ethical guidelines and practical implementation. As highlighted in Section 3.2, **?** found "complete ignorance" of ethical considerations in AI development within startup-like environments, often justified by a "prototype" mindset. This gap is further confirmed by **?**, which conducted a gap analysis comparing company practices to trustworthy AI requirements and found that novel requirements for societal well-being, diversity, non-discrimination, and fairness were largely unaddressed. Reinforcing this, **?**'s rapid review of Responsible AI (RAI) frameworks empirically demonstrated that only a small fraction offer practical tools, and most focus on early requirements elicitation, leaving critical later phases (design, development, testing, deployment) largely uncovered. This highlights a severe lack of actionable guidance for embedding ethics throughout the entire SDLC. **?** further emphasizes this by showing that existing AI lifecycle models are inadequate for regulated domains, necessitating new stages for documentation, model monitoring, and risk assessment to ensure comprehensive compliance.

In conclusion, the journey towards AI-driven software development compliance is fraught with practical and ethical complexities. The overhead of validation, the computational demands of advanced AI, the challenge of formalizing compliance rules, and the inherent risks of hallucinations, bias, and security vulnerabilities necessitate continuous vigilance. Addressing these challenges requires not only technical advancements, such as more "SE-aware" AI models **?** and improved explainability **?**, but also robust ethical frameworks, clear accountability mechanisms, and a commitment to responsible deployment strategies that prioritize human oversight, data privacy, algorithmic fairness, and comprehensive governance. The current state of practice, marked by the neglect of ethics in real-world projects and the lack of practical tools, underscores the urgent need for actionable methods and tools to bridge the gap between ethical principles and practical software engineering.

7.4 Future Research Directions

The evolving landscape of AI for software development compliance presents numerous promising avenues for future research. While significant progress has been made in automating compliance checks and integrating AI into various stages of the Software Development Life Cycle (SDLC), the ultimate goal remains the creation of inherently compliant, trustworthy, and adaptable AI systems that can operate across complex, real-world scenarios.

One critical direction involves developing more holistic, end-to-end AI-driven compliance frameworks that span the entire SDLC. Current research often focuses on isolated aspects, such as requirements analysis **?** or code analysis **?**. Future work should leverage platforms designed for generalist AI agents, such as OpenHands **?**, which provides a flexible, sandboxed environment and a programming language-based action space for agents to interact with diverse software development tasks. This foundation could be extended to host specialized compliance agents that continuously monitor, analyze, and enforce regulatory requirements from initial design to deployment and maintenance. For instance, the multi-agent collaborative framework demonstrated by MarsCode Agent **?** for automated bug fixing, which integrates LLMs with advanced code analysis techniques like Code Knowledge Graphs (CKG) and Language Server Protocols (LSP), offers a blueprint. A similar multi-agent approach could be adapted for compliance, where specialized agents handle legal interpretation, policy enforcement, and artifact verification across different SDLC phases, ensuring a unified and consistent compliance posture.

A major challenge for current AI systems, highlighted by recent benchmarks, is their limited generalization capabilities, particularly in multimodal and multi-language domains. Research must focus on enhancing multimodal and multi-language generalization for AI agents to accurately interpret diverse software artifacts and user interactions. The introduction of SWE-bench Multimodal **?** starkly reveals that existing AI systems struggle significantly with visual, user-facing JavaScript issues, demonstrating a critical gap in their ability to generalize beyond text-based, Python-centric tasks. Future compliance agents must be capable of understanding visual specifications, interpreting UI/UX designs for accessibility compliance, and analyzing codebases written in various programming languages, including those with complex asynchronous or visual components. This necessitates developing more flexible, language-agnostic agent architectures and multimodal perception tools that can effectively process images, videos, and diverse code structures to identify compliance deviations.

Furthermore, integrating advanced formal verification techniques with AI is crucial for building inherently compliant systems. While AI can identify potential issues, formal methods offer mathematical guarantees of correctness and adherence to specifications. Future research should explore how AI can assist in generating formal specifications from natural language requirements, automate the creation of proofs, or guide formal verification tools to focus on compliance-critical code sections. This hybrid approach would combine AI's scalability and pattern recognition with the rigor of formal methods, leading to more robust and legally defensible compliance assurances.

The scarcity of high-quality, compliance-specific datasets also presents a significant hurdle. Drawing inspiration from efforts like SWE-bench Multimodal **?** in creating domain-specific benchmarks, future research needs to focus on creating comprehensive datasets tailored to various regulatory frameworks (e.g., GDPR, HIPAA, industry standards). These datasets should include diverse examples of compliant and non-compliant code, documentation, and system behaviors, annotated with legal interpretations and contextual information. Such resources are essential for training and evaluating AI models that can accurately identify and mitigate compliance risks.

Improving the explainability and legal sufficiency of AI explanations is another paramount direction. For AI-driven compliance solutions to be adopted in regulated environments, their decisions must be transparent, auditable, and legally sound ?. Future research should develop novel Explainable AI (XAI) techniques that not only provide human-understandable rationales for compliance decisions but also link these explanations directly to specific legal clauses or regulatory requirements. This involves moving beyond mere feature importance to generating legally sufficient narratives that can withstand scrutiny from auditors and legal professionals.

Finally, exploring the socio-technical aspects of human-AI collaboration in compliance-critical tasks is vital. As AI agents become more autonomous, understanding how humans interact with and trust these systems, particularly when compliance is at stake, becomes crucial. Research should investigate optimal human-in-the-loop strategies, mechanisms for effective communication between human experts and AI agents, and the ethical implications of delegating compliance responsibilities to AI. This necessitates fostering interdisciplinary approaches that combine expertise from AI, software engineering, law, ethics, and social sciences to build inherently compliant and trustworthy AI systems that augment, rather than replace, human oversight in the future.

References

References

Marc G. Bellemare, Yavar Naddaf, J. Veness, et al. (2012). *The Arcade Learning Environment: An Evaluation Platform for General Agents*. Journal of Artificial Intelligence Research.

M. Cardoso, Wenqi Li, Richard Brown, et al. (2022). *MONAI: An open-source framework for deep learning in healthcare*. arXiv.org.

Saleema Amershi, Andrew Begel, C. Bird, et al. (2019). *Software Engineering for Machine Learning: A Case Study*. 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Carole-Jean Wu, R. Raghavendra, Udit Gupta, et al. (2021). *Sustainable AI: Environmental Implications, Challenges and Opportunities*. Conference on Machine Learning and Systems.

J. Laird (2012). *The Soar Cognitive Architecture*. Unpublished manuscript.

Miles Brundage, S. Avin, Jasmine Wang, et al. (2020). *Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims*. arXiv.org.

Jesse Dodge, Taylor Prewitt, Rémi Tachet des Combes, et al. (2022). *Measuring the Carbon Intensity of AI in Cloud Instances*. Conference on Fairness, Accountability and Transparency.

Neil Perry, Megha Srivastava, Deepak Kumar, et al. (2022). *Do Users Write More Insecure Code with AI Assistants?*. Conference on Computer and Communications Security.

Ruchi Puri, David S. Kung, G. Janssen, et al. (2021). *CodeNet: A Large-Scale AI for Code Dataset for Learning a Diversity of Coding Tasks*. NeurIPS Datasets and Benchmarks.

Ziqi Huang, Yang Shen, Jiayi Li, et al. (2021). *A Survey on AI-Driven Digital Twins in Industry 4.0: Smart Manufacturing and Advanced Robotics.* Italian National Conference on Sensors.

S. Bhat, and N. Huang (2021). *Big Data and AI Revolution in Precision Agriculture: Survey and Challenges.* IEEE Access.

Fei Wang, and A. Preininger (2019). *AI in Health: State of the Art, Challenges, and Future Directions.* Yearbook of Medical Informatics.

Tira Nur Fitria (2021). *Grammarly as AI-powered English Writing Assistant: Students' Alternative for Writing English.* Unpublished manuscript.

Aaron I F Poon, and J. Sung (2021). *Opening the black box of AI-Medicine.* Journal of Gastroenterology and Hepatology.

Lucy Ellen Lwakatare, Aiswarya Raj, J. Bosch, et al. (2019). *A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation.* International Conference on Agile Software Development.

J. Truby (2020). *Governing Artificial Intelligence to benefit the UN Sustainable Development Goals.* Sustainable Development.

Jieming Zhu, Shilin He, Pinjia He, et al. (2020). *Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics.* IEEE International Symposium on Software Reliability Engineering.

Haiyan Kong, Kang-Ting Wang, Xuejie Qiu, et al. (2022). *30 years of artificial intelligence (AI) research relating to the hospitality and tourism industry.* International Journal of Contemporary Hospitality Management.

Guoying Zhao, Yante Li, and Qianru Xu (2022). *From Emotion AI to Cognitive AI.* International Journal of Network Dynamics and Intelligence.

X. Ai, C. Allaire, N. Calace, et al. (2021). *A Common Tracking Software Project.* Computing and Software for Big Science.

Q. Lu, Liming Zhu, Xiwei Xu, et al. (2022). *Towards a Roadmap on Software Engineering for Responsible AI*. 2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN).

D. Widder, D. Nafus, Laura A. Dabbish, et al. (2022). *Limits and Possibilities for "Ethical AI" in Open Source: A Study of Deepfakes*. Conference on Fairness, Accountability and Transparency.

Mark Haakman, Luís Cruz, Hennie Huijgens, et al. (2020). *AI lifecycle models need to be revised*. Empirical Software Engineering.

F. Firouzi, Bahar Farahani, Mojtaba Barzegari, et al. (2022). *AI-Driven Data Monetization: The Other Face of Data in IoT-Based Smart and Connected Health*. IEEE Internet of Things Journal.

Vinod Kathail (2020). *Xilinx Vitis Unified Software Platform*. Symposium on Field Programmable Gate Arrays.

T. Sipola, Janne Alatalo, T. Kokkonen, et al. (2022). *Artificial Intelligence in the IoT Era: A Review of Edge AI Hardware and Software*. Conference of the Open Innovations Association.

C. Tanikawa, and T. Yamashiro (2021). *Development of novel artificial intelligence systems to predict facial morphology after orthognathic surgery and orthodontic treatment in Japanese patients*. Scientific Reports.

Hariharan Subramonyam, Jane Im, C. Seifert, et al. (2022). *Solving Separation-of-Concerns Problems in Collaborative Design of Human-AI Systems through Leaky Abstractions*. International Conference on Human Factors in Computing Systems.

Marco Barenkamp, Jonas Rebstadt, and Oliver Thomas (2020). *Applications of AI in classical software engineering*. AI Perspectives.

Danielle Gonzalez, Thomas Zimmermann, and Nachiappan Nagappan (2020). *The State* *of the ML-universe: 10 Years of Artificial Intelligence Machine Learning Software Development on GitHub*. IEEE Working Conference on Mining Software Repositories.

C. Richie (2022). *Environmentally sustainable development and use of artificial intelligence in health care*. Bioethics.

Ipek Ozkaya (2020). *What Is Really Different in Engineering AI-Enabled Systems?*. IEEE Software.

David C. Higgins, and V. Madai (2020). *From Bit to Bedside: A Practical Framework for Artificial Intelligence Product Development in Healthcare*. Advanced Intelligent Systems.

Piyush Khandelwal, Shiqi Zhang, Jivko Sinapov, et al. (2017). *BWIBots: A platform for bridging the gap between AI and human–robot interaction research*. Int. J. Robotics Res..

W. Hummer, Vinod Muthusamy, T. Rausch, et al. (2019). *ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI*. 2019 IEEE International Conference on Cloud Engineering (IC2E).

Saiqa Aleem, Luiz Fernando Capretz, and F. Ahmed (2016). *Game development software engineering process life cycle: a systematic review*. Journal of Software Engineering Research and Development.

André Steimers, and Moritz Schneider (2022). *Sources of Risk of AI Systems*. International Journal of Environmental Research and Public Health.

Nagadivya Balasubramaniam, Marjo Kauppinen, Kari Hiekkanen, et al. (2022). *Transparency and Explainability of AI Systems: Ethical Guidelines in Practice*. Requirements Engineering: Foundation for Software Quality.

J. Mezrich (2022). *Is Artificial Intelligence (AI) a Pipe Dream? Why Legal Issues Present Significant Hurdles to AI Autonomy.*. AJR. American journal of roentgenology.

Ki Hyun Tae, Yuji Roh, Young Hun Oh, et al. (2019). *Data Cleaning for Accurate, Fair, and Robust Models: A Big Data - AI Integration Approach.* DEEM@SIGMOD.

H. Belani, M. Vuković, and Z. Car (2019). *Requirements Engineering Challenges in Building AI-Based Complex Systems.* 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW).

José Carlos Sánchez Prieto, Juan Cruz-Benito, Roberto Therón, et al. (2020). *Assessed by Machines: Development of a TAM-Based Tool to Measure AI-based Assessment Acceptance Among Students.* Int. J. Interact. Multim. Artif. Intell..

Lucy Ellen Lwakatare, I. Crnkovic, and J. Bosch (2020). *DevOps for AI – Challenges in Development of AI-enabled Applications.* International Conference on Software, Telecommunications and Computer Networks.

Romina Eramo, V. Muttillo, Luca Berardinelli, et al. (2021). *AIDOaRt: AI-augmented Automation for DevOps, a Model-based Framework for Continuous Development in Cyber-Physical Systems.* Euromicro Symposium on Digital Systems Design.

William Tanguay, Philippe Acar, Benjamin Fine, et al. (2022). *Assessment of Radiology Artificial Intelligence Software: A Validation and Evaluation Framework.* Canadian Association of Radiologists journal = Journal l'Association canadienne des radiologistes.

Q. Lu, Liming Zhu, Xiwei Xu, et al. (2021). *Software engineering for Responsible AI: An empirical study and operationalised patterns.* 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

T. Leiner, E. Bennink, Christian P. Mol, et al. (2021). *Bringing AI to the clinic: blueprint for a vendor-neutral AI deployment infrastructure.* Insights into Imaging.

Samuli Laato, Teemu Birkstedt, Matti Mäntymäki, et al. (2022). *AI Governance in the System Development Life Cycle: Insights on Responsible Machine Learning Engineering.* 2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN).

Hans-Martin Heyn, E. Knauss, Amna Pir Muhammad, et al. (2021). *Requirement Engineering Challenges for AI-intense Systems Development.* Workshop on AI Engineering - Software Engineering for AI.

Mildred K. Cho (2021). *Rising to the challenge of bias in health care AI.* Nature Network Boston.

Ville Vakkuri, Kai-Kristian Kemell, Marianna Jantunen, et al. (2020). *"This is Just a Prototype": How Ethics Are Ignored in Software Startup-Like Environments.* International Conference on Agile Software Development.

Amos Lal, Johnny Dang, C. Nabzdyk, et al. (2022). *Regulatory oversight and ethical concerns surrounding software as medical device (SaMD) and digital twin technology in healthcare.* Annals of Translational Medicine.

N. Janbi, Rashid Mehmood, Iyad A. Katib, et al. (2022). *Imtidad: A Reference Architecture and a Case Study on Developing Distributed AI Services for Skin Disease Diagnosis over Cloud, Fog and Edge.* Italian National Conference on Sensors.

B. McCrindle, K. Zukotynski, Thomas E. Doyle, et al. (2021). *A Radiology-focused Review of Predictive Uncertainty for AI Interpretability in Computer-assisted Segmentation..* Radiology: Artificial Intelligence.

Meenu Mary John, H. H. Olsson, and Jan Bosch (2022). *Towards an AI-driven business development framework: A multi-case study.* J. Softw. Evol. Process..

Veronica Magni, M. Interlenghi, A. Cozzi, et al. (2022). *Development and Validation of an AI-driven Mammographic Breast Density Classification Tool Based on Radiologist Consensus..* Radiology: Artificial Intelligence.

T. Ishikawa, Masato Hayakawa, Hirotaka Suzuki, et al. (2022). *Development of a Novel Evaluation Method for Endoscopic Ultrasound-Guided Fine-Needle Biopsy in Pancreatic Diseases Using Artificial Intelligence.* Diagnostics.

Stavros Kalapothas, Georgios Flamis, and P. Kitsos (2022). *Efficient Edge-AI Application Deployment for FPGAs*. Inf..

Siddhant Singh, and H. Thakur (2020). *Survey of Various AI Chatbots Based on Technology Used*. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).

Izabela Rojek, D. Mikołajewski, M. Macko, et al. (2021). *Optimization of Extrusion-Based 3D Printing Process Using Neural Networks for Sustainable Development*. Materials.

Sergio Moreschini, Francesco Lomio, David Hästbacka, et al. (2022). *MLOps for evolvable AI intensive software systems*. IEEE International Conference on Software Analysis, Evolution, and Reengineering.

Yongli Zhao, Boyuan Yan, Zhuotong Li, et al. (2019). *Coordination between control layer AI and on-board AI in optical transport networks [Invited]*. IEEEOSA Journal of Optical Communications and Networking.

Lukas Fischer, Lisa Ehrlinger, V. Geist, et al. (2020). *AI System Engineering - Key Challenges and Lessons Learned*. Machine Learning and Knowledge Extraction.

Alene K. Rhea, K. Markey, L. D'Arinzo, et al. (2022). *An external stability audit framework to test the validity of personality prediction in AI hiring*. Data mining and knowledge discovery.

Shih-Yeh Chen, Yu-Sheng Su, Yaochia Ku, et al. (2022). *Exploring the factors of students' intention to participate in AI software development*. Library hi tech.

B. M. A. Matsui, and Denise H. Goya (2022). *MLOps: A Guide to its Adoption in the Context of Responsible AI*. 2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI).

Hung-Chang Chen, Shin-Shi Tzeng, Yen-Chang Hsiao, et al. (2021). *Smartphone-Based Artificial Intelligence–Assisted Prediction for Eyelid Measurements: Algorithm Development and Observational Validation Study*. JMIR mHealth and uHealth.

Vijay Bhasker, Reddy Bhimanapati, Dr. Punit Goel, et al. (2021). *Effective Use of AI-Driven Third-Party Frameworks in Mobile Apps.* Innovative Research Thoughts.

Ramon Correa, M. Shaan, H. Trivedi, et al. (2022). *A Systematic Review of 'Fair' AI Model Development for Image Classification and Prediction.* Journal of Medical and Biological Engineering.

Adina Aniculaesei, Jörg Grieser, A. Rausch, et al. (2018). *Toward a Holistic Software Systems Engineering Approach for Dependable Autonomous Systems.* 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS).

Quan Ze Chen, Tobias Schnabel, Besmira Nushi, et al. (2022). *HINT: Integration Testing for AI-based features with Humans in the Loop.* International Conference on Intelligent User Interfaces.

M. Meesters, P. Heck, and A. Serebrenik (2022). *What Is an AI Engineer? An Empirical Analysis of Job Ads in The Netherlands.* 2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN).

J. Bosch, H. H. Olsson, and I. Crnkovic (2018). *It takes three to tango: Requirement, outcome/data, and AI driven development.* Workshop on Software-intensive Business.

L. M. Cysneiros, Majid Raffi, and Julio Cesar Sampaio do Prado Leite (2018). *Software Transparency as a Key Requirement for Self-Driving Cars.* IEEE International Requirements Engineering Conference.

Mahesha Pandit, Deepali Gupta, Divya Anand, et al. (2022). *Towards Design and Feasibility Analysis of DePaaS: AI Based Global Unified Software Defect Prediction Framework.* Applied Sciences.

Kaihua Liu, S. Reddivari, and Kalyan Reddivari (2022). *Artificial Intelligence in Software Requirements Engineering: State-of-the-Art.* IEEE International Conference on Information Reuse and Integration.

L. Raamesh, S. Jothi, and S. Radhika (2022). *Enhancing Software Reliability and Fault Detection Using Hybrid Brainstorm Optimization-Based LSTM Model*. Journal of the Institution of Electronics and Telecommunication Engineers.

Ville Vakkuri, Kai-Kristian Kemell, Joel Tolvanen, et al. (2022). *How Do Software Companies Deal with Artificial Intelligence Ethics? A Gap Analysis*. International Conference on Evaluation Assessment in Software Engineering.

Elena Planas, Gwendal Daniel, Marco Brambilla, et al. (2021). *Towards a model-driven approach for multiexperience AI-based user interfaces*. Journal of Software and Systems Modeling.

Valentina Golendukhina, Valentina Lenarduzzi, and M. Felderer (2022). *What is Software Quality for AI Engineers? Towards a Thinning of the Fog*. 2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN).

Fangmin Xu, Fan Yang, Shijian Bao, et al. (2019). *DQN Inspired Joint Computing and Caching Resource Allocation Approach for Software Defined Information-Centric Internet of Things Network*. IEEE Access.

Jiachen Song, Linan Zhang, Jinglei Yu, et al. (2022). *Paving the Way for Novices: How to Teach AI for K-12 Education in China*. AAAI Conference on Artificial Intelligence.

Ekaterina A. Moroz, Vladimir O. Grizkevich, and I. M. Novozhilov (2022). *The Potential of Artificial Intelligence as a Method of Software Developer's Productivity Improvement*. 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus).

Mohamed Abdullahi Ali, Ng Keng Yap, A. Ghani, et al. (2022). *A Systematic Mapping of Quality Models for AI Systems, Software and Components*. Applied Sciences.

Chia-Ching Yuan, Cheng-Hsuan Li, and Chin-Cheng Peng (2021). *Development of mobile interactive courses based on an artificial intelligence chatbot on the communication software LINE*. Interactive Learning Environments.

Rajesh Kulkarni, and P. Padmanabham (2017). *Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration.* IET Software.

Karthikeyan Sankaralingam, Tony Nowatzki, Vinay Gangadhar, et al. (2022). *The Mozart reuse exposed dataflow processor for AI and beyond: industrial product.* International Symposium on Computer Architecture.

Mitsuru Yuba, and Kiyotaka Iwasaki (2022). *Systematic analysis of the test design and performance of AI/ML-based medical devices approved for triage/detection/diagnosis in the USA and Japan.* Scientific Reports.

Kishore Sugali, Chris Sprunger, and Venkata N. Inukollu (2021). *Software Testing: Issues and Challenges of Artificial Intelligence Machine Learning.* International Journal of Artificial Intelligence Applications.

Yan Zhao, Ning Hu, Yue Zhao, et al. (2021). *A secure and flexible edge computing scheme for AI-driven industrial IoT.* Cluster Computing.

Nadja Damij, and S. Bhattacharya (2022). *The Role of AI Chatbots in Mental Health Related Public Services in a (Post)Pandemic World: A Review and Future Research Agenda.* 2022 IEEE Technology and Engineering Management Conference (TEMSCON EUROPE).

Aastha Pant, Rashina Hoda, C. Tantithamthavorn, et al. (2022). *Ethics in AI through the practitioner's view: a grounded theory literature review.* Empirical Software Engineering.

Yong Cao, Rui Wang, Min Chen, et al. (2020). *AI Agent in Software-Defined Network: Agent-Based Network Service Prediction and Wireless Resource Scheduling Optimization.* IEEE Internet of Things Journal.

Jie Zhu, Leye Wang, and Xiao Han (2022). *Safety and Performance, Why not Both? Bi-Objective Optimized Model Compression toward AI Software Deployment.* International Conference on Automated Software Engineering.

L. M. Cysneiros, and Julio Cesar Sampaio do Prado Leite (2020). *Non-Functional Requirements Orienting the Development of Socially Responsible Software.* BP-MDS/EMMSAD@CAiSE.

Irene Nandutu, M. Atemkeng, and Patrice Okouma (2021). *Integrating AI ethics in wildlife conservation AI systems in South Africa: a review, challenges, and future research agenda.* Ai Society.

A. B. Tosun, F. Pullara, M. Becich, et al. (2020). *HistoMapr™: An Explainable AI (xAI) Platform for Computational Pathology Solutions.* AI and ML for Digital Pathology.

Isa Maleki, A. Ghaffari, and Mohammad Masdari (2014). *A New Approach for Software Cost Estimation with Hybrid Genetic Algorithm and Ant Colony Optimization.* Unpublished manuscript.

Bram Adams, and Foutse Khomh (2020). *The Diversity Crisis of Software Engineering for Artificial Intelligence.* IEEE Software.

K. Yeung, A. Howes, and Ganna Pogrebna (2020). *AI Governance by Human Rights–Centered Design, Deliberation, and Oversight.* Unpublished manuscript.

Ville Vakkuri, Marianna Jantunen, Erika Halme, et al. (2021). *Time for AI (Ethics) Maturity Model Is Now.* SafeAI@AAAI.

Hashem Alyami, M. Nadeem, Wael Alosaimi, et al. (2022). *Analyzing the Data of Software Security Life-Span: Quantum Computing Era.* Intelligent Automation and Soft Computing.

Enrique Torres-Sánchez, Jesús Alastruey-Benedé, and Enrique F. Torres Moreno (2020). *Developing an AI IoT application with open software on a RISC-V SoC.* Conference on Design of Circuits and Integrated Systems.

Angela Barriga, Adrian Rutle, and Rogardt Heldal (2022). *AI-powered model repair: an experience report—lessons learned, challenges, and opportunities.* Journal of Software and Systems Modeling.

Conrad Sanderson, Qinghua Lu, David M. Douglas, et al. (2022). *Towards Implementing Responsible AI.* 2022 IEEE International Conference on Big Data (Big Data).

M. d'Inverno, and J. Mccormack (2015). *Heroic versus Collaborative AI for the Arts.* International Joint Conference on Artificial Intelligence.

Markus Borg (2021). *Agility in Software 2.0 - Notebook Interfaces and MLOps with Buttresses and Rebars.* International Conference on Lean and Agile Software Development.

L. Surya (2021). *An exploratory study of AI and Big Data, and it's future in the United States.* Unpublished manuscript.

Takao Ito, M. Tanaka, Masako Shin, et al. (2021). *THE ONLINE PBL (PROJECT-BASED LEARNING) EDUCATION SYSTEM USING AI (ARTIFICIAL INTELLIGENCE).* DS 110: Proceedings of the 23rd International Conference on Engineering and Product Design Education (EPDE 2021).

S. Kavitha, J. V. Anchitaalagammai, S. Nirmala, et al. (2021). *Current Trends in Integrating the Internet of Things Into Software Engineering Practices.* Research Anthology on Artificial Intelligence Applications in Security.

Garima Suman, A. Panda, P. Korfiatis, et al. (2020). *Development of a volumetric pancreas segmentation CT dataset for AI applications through trained technologists: a study during the COVID 19 containment phase.* Abdominal Radiology.

Huakun Huang, M. Ogbodo, Zhishang Wang, et al. (2021). *Smart Energy Management System based on Reconfigurable AI Chip and Electrical Vehicles.* International Conference on Big Data and Smart Computing.

Divanshi Priyadarshni Wangoo (2018). *Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design.* International Conference on Computing, Communication and Automation.

A. Gusev, S. Morozov, V. A. Kutichev, et al. (2021). *Legal regulation of artificial intelligence software in healthcare in the Russian Federation.* Medical Technologies. Assessment and Choice.

Johan Aronsson, Philip Lu, D. Strüber, et al. (2020). *A maturity assessment framework for conversational AI development platforms.* ACM Symposium on Applied Computing.

Ville Vakkuri, Kai-Kristian Kemell, and P. Abrahamsson (2019). *Ethically Aligned Design: An Empirical Evaluation of the RESOLVEDD-Strategy in Software and Systems Development Context.* EUROMICRO Conference on Software Engineering and Advanced Applications.

Mudita, and D. Gupta (2020). *The Aspects of Artificial Intelligence in Software Engineering.* Unpublished manuscript.

Satyabrata Pradhan, Venky Nanniyur, and Pavan K. Vissapragada (2020). *On the Defect Prediction for Large Scale Software Systems – From Defect Density to Machine Learning.* International Conference on Software Quality, Reliability and Security.

Marcel Hauck, Rüdiger Machhamer, Levin Czenkusch, et al. (2019). *Node and Block-Based Development Tools for Distributed Systems With AI Applications.* IEEE Access.

J. Olszewska (2019). *D7-R4: Software Development Life-Cycle for Intelligent Vision Systems.* International Conference on Knowledge Engineering and Ontology Development.

Unknown Authors (2019). *Artificial Intelligence in Software Engineering: Current Developments and Future Prospects.* Unpublished manuscript.

Tunio Muhammad Zahid, Haiyong Luo, Cong Wang, et al. (2018). *Crowdsourcing Software Development: Task Assignment Using PDDL Artificial Intelligence Planning.* Journal of Information Processing Systems.

S. Parsaeefard, Iman Tabrizian, and Alberto Leon-Garcia (2019). *Artificial Intelligence as a Service (AI-aaS) on Software-Defined Infrastructure.* IEEE Conference on Standards for Communications and Networking.

Unknown Authors (2019). *Business Process Reengineering: A Scope of Automation in Software Project Management using Artificial Intelligence.* International Journal of Engineering and Advanced Technology.

Bhagyashree W. Sorte, P. P. Joshi, and V. Jagtap (2015). *Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review.* Unpublished manuscript.

Ekrem Kocaguneli, Ayse Tosun Misirli, and A. Bener (2010). *AI-Based Models for Software Effort Estimation.* EUROMICRO Conference on Software Engineering and Advanced Applications.

A. Ricci (2011). *Agent-Oriented Computing : Agents as a Paradigm for Computer Programming and Software Development.* Unpublished manuscript.

F. S. Gharehchopogh, Isa Maleki, Amin Kamalinia, et al. (2014). *Artificial bee colony based constructive cost model for software cost estimation.* Unpublished manuscript.

K. Shankari, and Dr.R. Thirumalaiselvi (2014). *A Survey on Using Artificial Intelligence Techniques in the Software Development Process.* Unpublished manuscript.

S. Jayavel, S. Arumugam, B. Singh, et al. (2013). *USE OF ARTIFICIAL INTELLIGENCE IN AUTOMATION OF SEQUENTIAL STEPS OF SOFTWARE DEVELOPMENT / PRODUCTION.* Unpublished manuscript.

Xingyao Wang, Boxuan Li, Yufan Song, et al. (2024). *OpenHands: An Open Platform for AI Software Developers as Generalist Agents.* International Conference on Learning Representations.

Sida Peng, Eirini Kalliamvakou, Peter Cihon, et al. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.* arXiv.org.

Zehang Deng, Yongjian Guo, Changzhou Han, et al. (2024). *AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways.* ACM Computing Surveys.

Jaakko Sauvola, Sasu Tarkoma, Mika Klemettinen, et al. (2024). *Future of software development with generative AI*. International Conference on Automated Software Engineering.

John Yang, Carlos E. Jimenez, Alex L. Zhang, et al. (2024). *SWE-bench Multimodal: Do AI Systems Generalize to Visual Software Domains?*. arXiv.org.

Daniel Russo (2023). *Navigating the Complexity of Generative AI Adoption in Software Engineering*. ACM Transactions on Software Engineering and Methodology.

C. Ebert, Panos Louridas, and C. Ebert (2023). *Generative AI for Software Practitioners*. IEEE Software.

Sanka Rasnayaka, Guanlin Wang, Ridwan Shariffdeen, et al. (2024). *An Empirical Study on Usage and Perceptions of LLMs in a Software Engineering Project*. 2024 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code).

Yizhou Liu, Pengfei Gao, Xinchen Wang, et al. (2024). *MarsCode Agent: AI-native Automated Bug Fixing*. arXiv.org.

Asha Rajbhoj, Akanksha Somase, Piyush Kulkarni, et al. (2024). *Accelerating Software Development Using Generative AI: ChatGPT Case Study*. International Symposium on Electronic Commerce.

Nnaemeka Valentine Eziamaka, Theodore Narku Odonkor, and Adetola Adewale Akinsulire (2024). *AI-Driven accessibility: Transformative software solutions for empowering individuals with disabilities*. International journal of applied research in social sciences.

Olga Petrovska, Lee Clift, Faron Moller, et al. (2024). *Incorporating Generative AI into Software Development Education*. Conference on Computing Education Practice.

Tobias Clement, Nils Kemmerzell, Mohamed Abdelaal, et al. (2023). *XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process*. Machine Learning and Knowledge Extraction.

Daniel Ajiga, Patrick Azuka Okeleke, Samuel Olaoluwa Folorunsho, et al. (2024). *Enhancing software development practices with AI insights in high-tech companies.* Computer Science amp; IT Research Journal.

M. Kuhail, S. Mathew, Ashraf Khalil, et al. (2024). *"Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools.* Science of Computer Programming.

Osinachi Deborah Segun-Falade, Olajide Soji Osundare, Wagobera Edgar Kedi, et al. (2024). *Developing innovative software solutions for effective energy management systems in industry.* Engineering Science amp; Technology Journal.

J. Klemmer, Stefan Albert Horstmann, Nikhil Patnaik, et al. (2024). *Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns.* Conference on Computer and Communications Security.

Mariana Coutinho, Lorena Marques, Anderson Santos, et al. (2024). *The Role of Generative AI in Software Development Productivity: A Pilot Case Study.* AIware.

A. I. Champa, Md. Fazle Rabbi, Costain Nachuma, et al. (2024). *ChatGPT in Action: Analyzing Its Use in Software Development.* IEEE Working Conference on Mining Software Repositories.

F. Laganá, Danilo Pratticò, G. Angiulli, et al. (2024). *Development of an Integrated System of sEMG Signal Acquisition, Processing, and Analysis with AI Techniques.* Signals.

Jun Liu, Cong Wang, Zile Liu, et al. (2024). *A bibliometric analysis of generative AI in education: current status and development.* Asia Pacific Journal of Education.

Michele Tufano, Anisha Agarwal, Jinu Jang, et al. (2024). *AutoDev: Automated AI-Driven Development.* arXiv.org.

M. Alenezi, and Mohammed Akour (2025). *AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions.* Applied Sciences.

Rasmus Ulfsnes, N. B. Moe, V. Stray, et al. (2024). *Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow.* arXiv.org.

Christopher Bull, and Ahmed Kharrufa (2023). *Generative Artificial Intelligence Assistants in Software Development Education: A Vision for Integrating Generative Artificial Intelligence Into Educational Practice, Not Instinctively Defending Against It.* IEEE Software.

Ahmed E. Hassan, G. Oliva, Dayi Lin, et al. (2024). *Rethinking Software Engineering in the Foundation Model Era: From Task-Driven AI Copilots to Goal-Driven AI Pair Programmers.* arXiv.org.

L. M. Amugongo, Alexander Kriebitz, Auxane Boch, et al. (2023). *Operationalising AI ethics through the agile software development lifecycle: a case study of AI-enabled mobile health applications.* AI and Ethics.

A. Bahi, Jihane Gharib, and Youssef Gahi (2024). *Integrating Generative AI for Advancing Agile Software Development and Mitigating Project Management Challenges.* International Journal of Advanced Computer Science and Applications.

Harrison Oke Ekpobimi, Regina Coelis Kandekere, and Adebamigbe Fasanmade (2024). *The future of software development: integrating AI and machine learning into front-end technologies.* Global Journal of Advanced Research and Reviews.

Y. Ouh, Tae Jin Kim, Woong Ju, et al. (2024). *Development and validation of artificial intelligence-based analysis software to support screening system of cervical intraepithelial neoplasia.* Scientific Reports.

Ruchika Pandey, Prabhat Singh, Raymond Wei, et al. (2024). *Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects.* arXiv.org.

Yanming Yang, Xing Hu, Zhipeng Gao, et al. (2024). *Federated Learning for Software Engineering: A Case Study of Code Clone Detection and Defect Prediction.* IEEE Transactions on Software Engineering.

Tobias Eisenreich, Sandro Speth, and Stefan Wagner (2024). *From Requirements to Architecture: An AI-Based Journey to Semi-Automatically Generate Software Architectures.* 2024 IEEE/ACM International Workshop on Designing Software (Designing).

Ali Husnain, Ghaith Alomari, and Ayesha Saeed (2024). *Ai-driven Integrated Hardware and Software Solution for EEG-based Detection of Depression and Anxiety.* International Journal For Multidisciplinary Research.

A. Odeh, Nada Odeh, and Abdul Salam Mohammed (2024). *A Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions.* TEM Journal.

Kanishk Dwivedi, Majid Haghparast, and T. Mikkonen (2024). *Quantum software engineering and quantum software development lifecycle: a survey.* Cluster Computing.

Michael Fu, Jirat Pasuksmit, and C. Tantithamthavorn (2024). *AI for DevSecOps: A Landscape and Future Opportunities.* ACM Transactions on Software Engineering and Methodology.

Ahmed E. Hassan, G. Oliva, Dayi Lin, et al. (2024). *Towards AI-Native Software Engineering (SE 3.0): A Vision and a Challenge Roadmap.* arXiv.org.

Thomas Dohmke, M. Iansiti, and Gregory L. Richards (2023). *Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle.* Unpublished manuscript.

Hakan Amasya, P. Jaju, M. Ezhov, et al. (2023). *Development and validation of an artificial intelligence software for periodontal bone loss in panoramic imaging.* International journal of imaging systems and technology (Print).

Ipek Ozkaya (2023). *The Next Frontier in Software Development: AI-Augmented Software Development Processes.* IEEE Software.

Amanda Fernandez, and Kimberly A. Cornell (2024). *CS1 with a Side of AI: Teaching*

Software Verification for Secure Code in the Era of Generative AI. Technical Symposium on Computer Science Education.

U. Durrani, Mustafa Akpinar, M. Fatih Adak, et al. (2024). *A Decade of Progress: A Systematic Literature Review on the Integration of AI in Software Engineering Phases and Activities (2013-2023).* IEEE Access.

Vita Santa Barletta, D. Caivano, Domenico Gigante, et al. (2023). *A Rapid Review of Responsible AI frameworks: How to guide the development of ethical AI.* International Conference on Evaluation Assessment in Software Engineering.

Valerio Terragni, Annie Vella, Partha S. Roop, et al. (2025). *The Future of AI-Driven Software Engineering.* ACM Transactions on Software Engineering and Methodology.

Sai Zhang, Zhenchang Xing, Ronghui Guo, et al. (2024). *Empowering Agile-Based Generative Software Development through Human-AI Teamwork.* ACM Transactions on Software Engineering and Methodology.

Andrés Neyem, Luis A. González, M. Mendoza, et al. (2024). *Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development.* IEEE Transactions on Learning Technologies.

Venkata Suresh Babu Chilluri, Sanjay Oli, Chandani Sharma, et al. (2025). *Design and Implementation of a User Centered Design Framework for Software Development.* 2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT).

Valerio Terragni, Partha S. Roop, and Kelly Blincoe (2024). *The Future of Software Engineering in an AI-Driven World.* arXiv.org.

Abdul Sajid Mohammed, Venkata Ramana Saddi, Santhosh Kumar Gopal, et al. (2024). *AI-Driven Continuous Integration and Continuous Deployment in Software Engineering.* International Conference on Database Theory.

Mateusz Dolata, Norbert Lange, and Gerhard Schwabe (2024). *Development in Times of Hype: How Freelancers Explore Generative AI?*. International Conference on Software Engineering.

Rohith Pudari, and Neil A. Ernst (2023). *From Copilot to Pilot: Towards AI Supported Software Development.* arXiv.org.

Santhosh Bussa (2024). *Evolution of Data Engineering in Modern Software Development.* Journal of Sustainable Solutions.

Tao Xiao, Hideaki Hata, Christoph Treude, et al. (2024). *Generative AI for Pull Request Descriptions: Adoption, Impact, and Developer Interventions.* Proc. ACM Softw. Eng..

Hazem W. Marar (2024). *Advancements in software engineering using AI.* Computer Software and Media Applications.

Ahmad Suliman Alnsour, O. Kanaan, Maimar Salah, et al. (2024). *The impact of implementing AI in recruitment on human resource management efficiency and organizational development effectiveness.* Journal of Infrastructure Policy and Development.

Unknown Authors (2024). *Generative AI for Effective Software Development.* Unpublished manuscript.

Peiyun Zhang, Song Ding, and Qianchuan Zhao (2023). *Exploiting Blockchain to Make AI Trustworthy: A Software Development Lifecycle View.* ACM Computing Surveys.

Muhammad Waseem, Teerath Das, Aakash Ahmad, et al. (2023). *ChatGPT as a Software Development Bot: A Project-Based Study.* International Conference on Evaluation of Novel Approaches to Software Engineering.

Peter Kokol (2024). *The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature.* Inf..

Danissa V. Rodriguez, K. Lawrence, Javier González, et al. (2023). *Leveraging Generative AI Tools to Support the Development of Digital Solutions in Health Care Research: Case Study.* JMIR Human Factors.

Yuchen Wang, Shangxin Guo, and Chee Wei Tan (2025). *From Code Generation to Software Testing: AI Copilot With Context-Based Retrieval-Augmented Generation.* IEEE Software.

Ameya Shastri Pothukuchi, Lakshmi Vasuda Kota, and Vinay Mallikarjunaradhya (2023). *IMPACT OF GENERATIVE AI ON THE SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC).* Unpublished manuscript.

Vasilka Saklamaeva, and Luka Pavlič (2023). *The Potential of AI-Driven Assistants in Scaled Agile Software Development.* Applied Sciences.

Panchanan Nath, Jaya Rani Mushahary, Ujjal Roy, et al. (2023). *AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development.* Computers electrical engineering.

Ze Shi Li, Nowshin Nawar Arony, Ahmed Musa Awon, et al. (2024). *AI Tool Use and Adoption in Software Development by Individuals and Organizations: A Grounded Theory Study.* arXiv.org.

Fangchen Song, Ashish Agarwal, and Wen Wen (2024). *The Impact of Generative AI on Collaborative Open-Source Software Development: Evidence from GitHub Copilot.* Social Science Research Network.

Xu-Kai Ma, Yan Yu, Tao Huang, et al. (2024). *Bioinformatics software development: Principles and future directions.* The Innovation Life.

S. M. Islam, Md Shadikul Bari, and Ankur Sarkar (2024). *Transforming Software Testing in the US: Generative AI Models for Realistic User Simulation.* Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023.

Jules White (2024). *Building Living Software Systems with Generative Agentic AI.* arXiv.org.

Mohammad Baqar, and Rajat Khanda (2024). *The Future of Software Testing: AI-Powered Test Case Generation and Validation.* Lecture Notes in Networks and Systems.

Mohammed Majid Bakhsh, Md Shaikat Alam Joy, and Gazi Touhidul Alam (2024). *Revolutionizing BA-QA Team Dynamics: AI-Driven Collaboration Platforms for Accelerated Software Quality in the US Market.* Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023.

Talia Gershon, Seetharami R. Seelam, Brian Belgodere, et al. (2024). *The infrastructure powering IBM's Gen AI model development.* arXiv.org.

N. Parikh (2023). *Empowering Business Transformation: The Positive Impact and Ethical Considerations of Generative AI in Software Product Management - A Systematic Literature Review.* arXiv.org.

Victoria Jackson, Bogdan Vasilescu, Daniel Russo, et al. (2024). *Creativity, Generative AI, and Software Development: A Research Agenda.* arXiv.org.

Peng Zhang, Yong Xiao, Yingyu Li, et al. (2023). *Toward Net-Zero Carbon Emissions in Network AI for 6G and Beyond.* IEEE Communications Magazine.

Zhihao Lin, Wei Ma, Tao Lin, et al. (2024). *Open Source AI-based SE Tools: Opportunities and Challenges of Collaborative Software Learning.* ACM Transactions on Software Engineering and Methodology.

M. T. Baldassarre, Domenico Gigante, Marcos Kalinowski, et al. (2024). *POLARIS: A framework to guide the development of Trustworthy AI systems.* 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN).

Zhenchang Xing, Qing Huang, Yu Cheng, et al. (2023). *Prompt Sapper: LLM-Empowered Software Engineering Infrastructure for AI-Native Services.* arXiv.org.

Andrés Neyem, Juan Pablo Sandoval Alcocer, M. Mendoza, et al. (2024). *Exploring the Impact of Generative AI for StandUp Report Recommendations in Software Capstone Project Development.* Technical Symposium on Computer Science Education.

Victoria Jackson, Bogdan Vasilescu, Daniel Russo, et al. (2024). *The Impact of Generative*

AI on Creativity in Software Development: A Research Agenda. ACM Transactions on Software Engineering and Methodology.

Ipek Ozkaya (2023). *Can Architecture Knowledge Guide Software Development With Generative AI?.* IEEE Software.

Malik Abdul Sami, Muhammad Waseem, Zeeshan Rasheed, et al. (2024). *Experimenting with Multi-Agent Software Development: Towards a Unified Platform.* arXiv.org.

Jordi Cabot, R. Clarisó, and T. Menzies (2023). *Low Code for Smart Software Development.* IEEE Software.

Santhosh Bussa (2023). *Artificial Intelligence in Quality Assurance for Software Systems.* Stallion Journal for Multidisciplinary Associated Research Studies.

Benjamin Klieger, Charis Charitsis, Miroslav Suzara, et al. (2024). *ChatCollab: Exploring Collaboration Between Humans and AI Agents in Software Teams.* arXiv.org.

Rohan Padhye (2024). *Software Engineering Methods for AI-Driven Deductive Legal Reasoning.* SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software.

Jie Zhu, Leye Wang, Xiao Han, et al. (2024). *Safety and Performance, Why Not Both? Bi-Objective Optimized Model Compression Against Heterogeneous Attacks Toward AI Software Deployment.* IEEE Transactions on Software Engineering.

Elise Paradis, Kate Grey, Quinn Madison, et al. (2024). *How Much Does AI Impact Development Speed? an Enterprise-Based Randomized Controlled Trial.* 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Muhammad Hamza, Dominik Siemon, M. Akbar, et al. (2023). *Human-AI Collaboration in Software Engineering: Lessons Learned from a Hands-On Workshop.* 2024 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB).

Vahit Bayrı, and Ece Demirel (2023). *AI-Powered Software Testing: The Impact of Large Language Models on Testing Methodologies.* 2023 4th International Informatics and Software Engineering Conference (IISEC).

Simon Rädler, Luca Berardinelli, Karolin Winter, et al. (2023). *Bridging MDE and AI: a systematic review of domain-specific languages and model-driven practices in AI software systems engineering.* Journal of Software and Systems Modeling.

Aarti (2024). *Generative Ai in Software Development : an Overview and Evaluation of Modern Coding Tools.* International Journal For Multidisciplinary Research.

Matteo Esposito, Xiaozhou Li, Sergio Moreschini, et al. (2025). *Generative AI for Software Architecture. Applications, Trends, Challenges, and Future Directions.* Journal of Systems and Software.

Ilche Georgievski (2023). *Conceptualising Software Development Lifecycle for Engineering AI Planning Systems.* 2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN).

Jamie Gorson Benario, Jenn Marroquin, Monica M. Chan, et al. (2025). *Unlocking Potential with Generative AI Instruction: Investigating Mid-level Software Development Student Perceptions, Behavior, and Adoption.* Technical Symposium on Computer Science Education.

Yatin Bajaj, and Manoj Samal (2023). *Accelerating Software Quality: Unleashing the Power of Generative AI for Automated Test-Case Generation and Bug Identification.* International Journal for Research in Applied Science and Engineering Technology.

Z. Özpolat, Özal Yıldırım, and M. Karabatak (2023). *Artificial Intelligence-Based Tools in Software Development Processes: Application of ChatGPT.* European Journal of Technic.

Akhil Raj, and Ridhi Deora (2025). *AI and ML Powered Feature Prioritization in Software Product Development.* International Journal of Data Mining amp; Knowledge Management Process.

Satish Chandra (2024). *AI in Software Engineering at Google: Progress and the Path Ahead (Invited Talk).* AIware.

Ronnie de Souza Santos, Felipe Fronchetti, Sávio Freire, et al. (2024). *Software Fairness Debt: Building a Research Agenda for Addressing Bias in AI Systems.* ACM Transactions on Software Engineering and Methodology.

Mario E. S. Simaremare, and Henry Edison (2024). *The State of Generative AI Adoption from Software Practitioners' Perspective: An Empirical Study.* EUROMICRO Conference on Software Engineering and Advanced Applications.

Vahid Garousi, Nithin Joy, Zafar Jafarov, et al. (2024). *AI-powered software testing tools: A systematic review and empirical assessment of their features and limitations.* Unpublished manuscript.

A. Oloruntoba, Å. Ingvar, M. Sashindranath, et al. (2024). *Examining labelling guidelines for AI-based software as a medical device: A review and analysis of dermatology mobile applications in Australia.* Australasian Journal of Dermatology.

Yu Han, Aaron Ceross, and Jeroen H. M. Bergmann (2023). *Regulatory Frameworks for AI-Enabled Medical Device Software in China: Comparative Analysis and Review of Implications for Global Manufacturer.* JMIR AI.

Agnia Sergeyuk, Ilya Zakharov, Ekaterina Koshchenko, et al. (2025). *Human-AI Experience in Integrated Development Environments: A Systematic Literature Review.* arXiv.org.

Sadra Sabouri, Philipp Eibl, Xinyi Zhou, et al. (2025). *Trust Dynamics in AI-Assisted Development: Definitions, Factors, and Implications.* International Conference on Software Engineering.

Palash Bera, Yves Wautelet, and G. Poels (2023). *On the Use of ChatGPT to Support Agile Software Development.* Agil-ISE@CAiSE.

Rahila Anwar, and Muhammad Bilal Bashir (2023). *A Systematic Literature Review of AI-Based Software Requirements Prioritization Techniques.* IEEE Access.

Antonella Borrelli, M. Pecoraro, F. del Giudice, et al. (2023). *Standardization of Body Composition Status in Patients with Advanced Urothelial Tumors: The Role of a CT-Based AI-Powered Software for the Assessment of Sarcopenia and Patient Outcome Correlation.* Cancers.

Xavier Franch, Andreas Jedlitschka, and Silverio Martínez-Fernández (2023). *A Requirements Engineering Perspective to AI-Based Systems Development: A Vision Paper.* Requirements Engineering: Foundation for Software Quality.

Muneera Bano, Didar Zowghi, V. Gervasi, et al. (2023). *AI for All: Operationalising Diversity and Inclusion Requirements for AI Systems.* arXiv.org.

Silverio Martínez-Fernández, Xavier Franch, and Francisco Durán (2023). *Towards green AI-based software systems: an architecture-centric approach (GAISSA).* EUROMICRO Conference on Software Engineering and Advanced Applications.

Illia Solohubov, Artur Moroz, M. Tiahunova, et al. (2023). *Accelerating software development with AI: exploring the impact of ChatGPT and GitHub Copilot.* CTE.

Seunghwan Cho, Ingyu Kim, Jinhan Kim, et al. (2023). *A Maturity Model for Trustworthy AI Software Development.* Applied Sciences.

M. Sendak, D. Vidal, Sylvia Trujillo, et al. (2023). *Editorial: Surfacing best practices for AI software development and integration in healthcare.* Frontiers in Digital Health.

Sajed Jalil (2023). *The Transformative Influence of LLMs on Software Development Developer Productivity.* 2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA).

P. Alves, and Bruno Pereira Cipriano (2023). *The centaur programmer - How Kasparov's Advanced Chess spans over to the software development of the future.* arXiv.org.

Benoît Combemale, J. Gray, and Bernhard Rumpe (2023). *Large language models as an "operating" system for software and systems modeling.* Journal of Software and Systems Modeling.

Justine Winata Purwoko, Tegar Abdullah, Budiman Wijaya, et al. (2023). *Analysis Chat-GPT Potential: Transforming Software Development with AI Chat Bots.* 2023 International Conference on Networking, Electrical Engineering, Computer Science, and Technology (IConNECT).

Hans-Martin Heyn, K. M. Habibullah, E. Knauss, et al. (2023). *Automotive Perception Software Development: An Empirical Investigation into Data, Annotation, and Ecosystem Challenges.* 2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN).

Josh Mahmood Ali (2023). *AI-driven software engineering.* Advances in Engineering Innovation.

A. Sorbello, S. A. Haque, Rashedul Hasan, et al. (2023). *Artificial Intelligence–Enabled Software Prototype to Inform Opioid Pharmacovigilance From Electronic Health Records: Development and Usability Study.* JMIR AI.

Filipe Calegario, V. Burégio, Francisco Erivaldo, et al. (2023). *Exploring the intersection of Generative AI and Software Development.* arXiv.org.

Beatriz Cabrero Daniel (2023). *AI for Agile development: a Meta-Analysis.* arXiv.org.

Nathaniel Crosley, R. E. Indrajit, and Erick Dazki (2023). *TOGAF Framework For an AI-enabled Software House.* SinkrOn.

Samarth Sikand, V. Sharma, Vikrant S. Kaulgud, et al. (2023). *Green AI Quotient: Assessing Greenness of AI-based software and the way forward.* International Conference on Automated Software Engineering.

Harman Yousif Ibrahim Khalid, and Najlaa Aldabagh (2024). *A Survey on the Latest Intrusion Detection Datasets for Software Defined Networking Environments.* Engineering, Technology amp; Applied Science Research.

Onur Polat, Saadin Oyucu, Muammer Türkoglu, et al. (2024). *Hybrid AI-Powered Real-Time Distributed Denial of Service Detection and Traffic Monitoring for Software-Defined-Based Vehicular Ad Hoc Networks: A New Paradigm for Securing Intelligent Transportation Networks.* Applied Sciences.

William F. Godoy, Pedro Valero-Lara, Keita Teranishi, et al. (2024). *Large language model evaluation for high-performance computing software development.* Concurrency and Computation.

Melissa Rochon, Judith Tanner, James Jurkiewicz, et al. (2024). *Wound imaging software and digital platform to assist review of surgical wounds using patient smartphones: The development and evaluation of artificial intelligence (WISDOM AI study).* PLoS ONE.

B. M. Xavier, Merim Dzaferagic, Irene Vila, et al. (2024). *Cross-Domain AI for Early Attack Detection and Defense Against Malicious Flows in O-RAN.* ICC 2024 - IEEE International Conference on Communications.

Sabbir M. Saleh, Ibrahim Mohammed Sayem, N. Madhavji, et al. (2024). *Advancing Software Security and Reliability in Cloud Platforms through AI-based Anomaly Detection.* CCSW@CCS.

Sunil GC, Arjun Upadhyay, and Xin Sun (2024). *Development of Software Interface for AI-Driven Weed Control in Robotic Vehicles, with Time-Based Evaluation in Indoor and Field Settings.* Smart Agricultural Technology.

Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, et al. (2024). *Developer Perspectives on Licensing and Copyright Issues Arising from Generative AI for Coding.* arXiv.org.

Jay U. Oswal, Harshil T. Kanakia, and Devvrat Suktel (2024). *Transforming Software Requirements into User Stories with GPT-3.5 -: An AI-Powered Approach.* 2024 2nd

International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT).

Rudolf Ramler, Michael Moser, Lukas Fischer, et al. (2024). *Industrial Experience Report on AI-Assisted Coding in Professional Software Development.* 2024 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code).

Ayyappa Sajja, Dheerender Thakur, and Aditya Mehra (2024). *Integrating Generative AI into the Software Development Lifecycle: Impacts on Code Quality and Maintenance.* International Journal of Science and Research Archive.

Ebtesam Al Haque, Chris Brown, Thomas D. Latoza, et al. (2024). *The Evolution of Information Seeking in Software Development: Understanding the Role and Impact of AI Assistants.* SIGSOFT FSE Companion.

Alex Gu, Naman Jain, Wen-Ding Li, et al. (2025). *Challenges and Paths Towards AI for Software Engineering.* arXiv.org.

C. Ebert, John Pravin Arockiasamy, Lennard Hettich, et al. (2024). *Hints for Generative AI Software Development.* IEEE Software.

Johannes Chen, and Jan Zacharias (2024). *Design Principles for Collaborative Generative AI Systems in Software Development.* International Conference on Design Science Research in Information Systems and Technology.

F. Pan, Yinglei Song, Long Wen, et al. (2025). *Automating Automotive Software Development: A Synergy of Generative AI and Formal Methods.* arXiv.org.

Zhi Chen, and Lingxiao Jiang (2024). *Evaluating Software Development Agents: Patch Patterns, Code Quality, and Issue Complexity in Real-World GitHub Scenarios.* IEEE International Conference on Software Analysis, Evolution, and Reengineering.

Negar Alizadeh, Boris Belchev, N. Saurabh, et al. (2024). *Language Models in Software Development Tasks: An Experimental Analysis of Energy and Accuracy.* IEEE Working Conference on Mining Software Repositories.

Ruiyin Li, Peng Liang, Yifei Wang, et al. (2025). *Unveiling the Role of ChatGPT in Software Development: Insights from Developer-ChatGPT Interactions on GitHub*. arXiv.org.

Christian Meske, Tobias Hermanns, Esther von der Weiden, et al. (2025). *Vibe Coding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda*. arXiv.org.

Xinyi Wang, Shaukat Ali, and Paolo Arcaini (2025). *Quantum Artificial Intelligence for Software Engineering: the Road Ahead*. arXiv.org.

Pramodya Samarakoon, Dinesh Asanka, Shantha Jayalal, et al. (2024). *Analyzing the Learning Effectiveness of Generative AI for Software Development for Undergraduates in Sri Lanka*. International Conference on Soft Computing and Software Engineering.

Vijayalakshmi Ramasamy, Suganya Ramamoorthy, G. Walia, et al. (2024). *Enhancing User Story Generation in Agile Software Development Through Open AI and Prompt Engineering*. Frontiers in Education Conference.

Shreyas Pangavhane, Gokul Raktate, Prasad Pariane, et al. (2024). *AI-Augmented Software Development: Boosting Efficiency and Quality*. 2024 International Conference on Decision Aid Sciences and Applications (DASA).

T. Bannon, and Phil Laplante (2024). *Generative AI in the Software Development Lifecycle*. Computer.

Kevin N. Shah, and Abhishek Trehan (2024). *STREAMLINING SOFTWARE DEVELOPMENT: A COMPARATIVE STUDY OF AI-DRIVEN AUTOMATION TOOLS IN MODERN TECH*. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING TECHNOLOGY.

Qing Wang, Junjie Wang, Mingyang Li, et al. (2024). *A Roadmap for Software Testing in Open-Collaborative and AI-Powered Era*. ACM Transactions on Software Engineering and Methodology.

Qing Wang, Junjie Wang, Mingyang Li, et al. (2024). *A Roadmap for Software Testing in Open Collaborative Development Environments.* arXiv.org.

Mohd Ariful Haque, Sunzida Siddique, Md. Mahfuzur Rahman, et al. (2025). *SOK: Exploring Hallucinations and Security Risks in AI-Assisted Software Development with Insights for LLM Deployment.* arXiv.org.

Uwe M. Borghoff, Mark Minas, and Jannis Schopp (2025). *Generative AI in Student Software Development Projects: A User Study on Experiences and Self-Assessment.* European Conference of Software Engineering Education.

Patricia de Oliveira Santos, Allan Chamon Figueiredo, Pedro Nuno Moura, et al. (2024). *Impacts of the Usage of Generative Artificial Intelligence on Software Development Process.* Brazilian Symposium on Information Systems.

Samarth Sikand, Kanchanjot Kaur Phokela, V. Sharma, et al. (2024). *How much SPACE do metrics have in GenAI assisted software development?.* International Symposium on Electronic Commerce.

Miroslaw Staron, S. Abrahão, Grace Lewis, et al. (2024). *Bringing Software Engineering Discipline to the Development of AI-Enabled Systems.* IEEE Software.

Yasushi Masumori, Soichiro Inoue, Yusuke Seino, et al. (2024). *Glottis Recognition Software Development Using Artificial Intelligence.* Cureus.

Yaya Gadjama Soureya, Amougou Ngoumou, J. M. Ngossaha, et al. (2025). *Adaptive software development: a comprehensive framework integrating artificial intelligence for sustainable evolution.* ~The œinternational Arab journal of information technology.

Guilherme Vaz Pereira, Victoria Jackson, R. Prikladnicki, et al. (2025). *Exploring GenAI in Software Development: Insights from a Case Study in a Large Brazilian Company.* 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Noshin Tasneem, H. Zulzalil, and Sa'adah Hassan (2025). *Enhancing Agile Software Development: A Systematic Literature Review of Requirement Prioritization and Reprioritization Techniques*. IEEE Access.

Ferenc Héjja, Tamás Bartók, Roy Dakroub, et al. (2024). *Generative AI for Productivity in Industry and Education*. International Conference on Complex Information Systems.

Xiaomeng Chai, Min Zhang, and Hua Tian (2024). *AI for Science: Practice from Baidu PaddlePaddle*. Portland International Conference on Management of Engineering and Technology.

M. M. Kholoosi, M. A. Babar, and Roland Croft (2024). *A Qualitative Study on Using ChatGPT for Software Security: Perception vs. Practicality*. International Conference on Trust, Privacy and Security in Intelligent Systems and Applications.

_1_the_evolving_l_and_scape_of_software_development__and__ai1.13The Evolving Landscape of Software Development and AIsubsection.1.1 _2_defining_ai_for_software_development_compliance1.25Defining AI for Software Development Compliancesubsection.1.2 _3_scope__and__structure_of_the_review1.37Scope and Structure of the Reviewsubsection.1.3 _capabilities_of_ai_in_software_engineering210Founda
Capabilities of AI in Software Engineeringsection.2 _1_ai_for_code:_datasets__and__core_capabil
for Code: Datasets and Core Capabilitiessubsection.2.1 _2_ai-assisted_development:_productivity
ai_collaboration2.213AI-Assisted Development: Productivity and Human-AI Collaborationsubsection.2.2 _3_autonomous_ai_agents__and__platforms_for_software_engineering2.316
AI Agents and Platforms for Software Engineeringsubsection.2.3 _imperative_for_trustworthy__ar
Imperative for Trustworthy and Responsible AI Developmentsection.3 _1_conceptualizing_trustworthy_ai__and__verifiable_claims3.118Conceptualizing Trustworthy AI and Verifiable Claimssubsection.3.1 _2_ethical_considerations__and__maturity_models_
Considerations and Maturity Models in AI Developmentsubsection.3.2 _3_environmental_sustainability_of_ai_systems3.323Environmental Sustainability of AI Systemssubsection.3.3 _ai_applications_for_automated_compliance_detection425Core AI Applications for Automated Compliance Detectionsection.4 _1_traditional_ai/ml_for_code__anc
AI/ML for Code and Artifact Analysissubsection.4.1 _2_ai_for_security__and__vulnerability_
for Security and Vulnerability Compliancesubsection.4.2 _3_ai_for_data_privacy__and__regula
for Data Privacy and Regulatory Adherencesubsection.4.3 _ai_applications_for_proactive_com
AI Applications for Proactive Compliance and Lifecycle Integrationsection.5 _1_leveraging_large_lang
to-code_mapping5.133Leveraging Large Language Models for Regulatory Interpretation and Policy-to-Code Mappingsubsection.5.1 _2_generative_ai_for_compliance-aware_design__and__code_generation5.235Generative AI for Compliance-Aware Design and Code Generationsubsection.5.2 _3_integrating_ai-driven_compliance_into_devops__a
AI-Driven Compliance into DevOps and Agile Workflowssubsection.5.3 _considerations_for_trus
driven_compliance_systems641Critical Considerations for Trustworthy AI-Driven Compliance Systemssection.6 _1_explainable_ai_(xai)_for_compliance_auditing__and__trust6.141Ex
AI (XAI) for Compliance Auditing and Trustsubsection.6.1 _2_blockchain_for_immutable_co