Agentic AI in Software Engineering: Practitioner Perspectives Across the Software Development Life Cycle

Muhammad Azeem Akbar
Software Engineering Department, Lappeenranta-Lahti University of Technology
Lappeenranta, Finland
azeem.akbar@lut.fi

Arif Ali Khan
M3S Research Unit, University of Oulu
Oulu, Finland
arif.khan@oulu.fi

Muhammad Hamza
Software Engineering Department, Lappeenranta-Lahti University of Technology
Lahti, Finland
muhammad.hamza@lut.fi

Abdul Razzaq
Department of Computer Science and Information Systems, University of Limerick
Lero, Ireland
abdul.razzaq@ul.ie

Abdullah Ghaffar
Department of Computer Science, COMSATS University
Islamabad, Pakistan
abdullahghaffar2003@gmail.com

Arash Hajikhani
VTT Technical Research Centre of Finland
Espoo, Finland
arash.hajikhani@vtt.fi

Abstract

Agentic artificial intelligence (Agentic AI) introduces autonomous decision-making and proactive problem-solving into software development, fundamentally reshaping each stage of the software development life cycle (SDLC). This study investigates the application and perceived impact of Agentic AI across the SDLC, from requirements engineering to software maintenance, through a qualitative, expert-driven approach. Twenty-one expert software engineers, AI specialists, and project managers from diverse industries participated in semi-structured interviews. Through thematic analysis, we identified key patterns in how Agentic AI influences requirements elicitation quality, system design adaptability, coding efficiency, automated testing coverage, and predictive maintenance. The findings reveal that Agentic AI not only augments existing workflows but also redefines traditional boundaries between SDLC phases by enabling continuous feedback loops, real-time optimization, and collaborative intelligence between human developers and AI agents. Despite challenges such as trust calibration, ethical concerns, and integration complexity, Agentic AI holds significant promise for accelerating development, reducing costs, and improving software quality. This paper offers practical and theoretical implications for researchers, practitioners, and tool developers aiming to leverage Agentic AI in software engineering.

Keywords

Agentic AI, Software Engineering, Challenges, Interview study

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Muhammad Azeem Akbar, Arif Ali Khan, Muhammad Hamza, Abdul Razzaq, Abdullah Ghaffar, and Arash Hajikhani. 2018. Agentic AI in Software Engineering: Practitioner Perspectives Across the Software Development Life Cycle. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Software engineering is undergoing a paradigm shift as artificial intelligence (AI) becomes increasingly integrated into the tools and processes that define modern software development. Traditional AI applications such as code completion, bug prediction, and requirements analysis have typically relied on static algorithms and human-initiated interactions [4, 10, 36]. In contrast, Agentic AI represents a new generation of AI systems capable of autonomous action, adaptive learning, and goal-oriented reasoning [22]. These agents can proactively initiate tasks, make context-aware decisions, and collaborate with human developers in a continuous, iterative manner.

The emergence of Agentic AI is fueled by advances in large language models (LLMs), reinforcement learning, multi-agent frameworks, and cloud-native orchestration [32]. This technological convergence enables software agents to not only respond to developer prompts but also to anticipate project needs, suggest architectural adjustments, perform code refactoring autonomously, and coordinate distributed development tasks [20, 24, 28]. As such, Agentic AI has the potential to impact every phase of the SDLC, transforming requirements elicitation into a predictive, analytics-driven process, accelerating design validation, generating optimized code, enhancing testing coverage, and enabling predictive maintenance [3, 32, 39].

However, the integration of Agentic AI into the SDLC brings new challenges in software engineering. Concerns around explainability, trustworthiness, regulatory compliance, and human-AI collaboration dynamics remain unresolved [27, 28]. Moreover, while there is growing interest in AI-assisted software engineering, the literature has yet to systematically examine the role of Agentic AI with its

autonomy and reasoning capabilities across the complete SDLC in real-world settings. Current studies largely focus on narrow applications, leaving a gap in understanding the broader, cross-phase impact of Agentic AI [9, 20, 24, 27, 28].

To address this gap, we conducted a qualitative study involving 21 experts with experience in software development, AI deployment, and project management. Through semi-structured interviews, we explored their perceptions, experiences, and concerns regarding the adoption of Agentic AI in each SDLC phase, from requirements engineering to maintenance. The study follows a thematic analysis approach, producing a rich dataset of practitioner insights, coded patterns, and emergent themes. The contributions of this paper are threefold:

- *Cross-SDLC synthesis:* We offer an in-depth, practitioner-grounded account of Agentic AI applicability across requirements, design, coding, testing, and maintenance.
- *Empirical insights:* From 21 expert interviews, we identify perceived benefits, risks, and enablers, and surface cross-phase themes (e.g., human–AI collaboration, trust, and governance).
- *Actionable guidance:* We derive adoption guidelines (readiness, tooling, and governance checklists) that can inform organizations piloting Agentic AI in their SDLC.

The remainder of this paper is organized as follows: Section 2 presents background literature on AI-assisted software development and Agentic AI concepts. Section 3 outlines the motivation for this research. Section 4 details our methodology, including participant selection and data analysis. Section 5 present our findings across the SDLC phases, whereas Section 6 discusses implications for research and practice, and addresses threats to validity. Finally, Section 8 concludes the paper and proposes avenues for future work.

2 Background

The software development life cycle (SDLC) is a structured framework for delivering software systems, encompassing sequential and iterative phases such as requirements engineering, design, coding, testing, deployment, and maintenance [19]. Over the past decades, advancements in tooling and methodologies ranging from model-driven engineering to DevOps have sought to improve efficiency, quality, and adaptability in these phases [8, 15]. However, as software complexity has grown, human-centered processes alone have struggled to meet increasing demands for speed, reliability, and continuous delivery.

Artificial intelligence (AI) in software engineering is not a new concept. Early applications included expert systems for debugging and search-based software engineering for optimization tasks [31]. More recently, machine learning (ML) approaches have been adopted for predictive defect detection, code recommendation, and automated test generation [3, 9]. Despite these advances, traditional AI in software engineering has primarily been reactive, responding to predefined queries or conditions and has lacked autonomy to operate proactively across the SDLC. To illustrate this evolution, Table 1 contrasts traditional AI, generative AI, and Agentic AI in the context of software engineering.

Agentic AI marks a significant shift from earlier paradigms. It describes AI systems endowed with the capacity for autonomous, goal-directed behavior capable of initiating actions [12], adapting to changing contexts, and coordinating tasks without constant human intervention [1, 14]. This capability is enabled by several converging technological trends, including:

- *Large Language Models (LLMs):* Systems such as GPT-4, Claude, and LLaMA, trained on vast corpora of code and natural language, can understand and generate both human-readable and machine-executable artifacts [33], bridging the gap between business requirements and technical implementation.
- *Multi-Agent Architectures:* Platforms like LangChain Agents, AutoGen, and Dapr-based orchestration allow multiple AI agents to collaborate, negotiate, and distribute tasks dynamically [11].
- *Reinforcement Learning and Planning Algorithms:* These enable AI to optimize development workflows over time, learning from outcomes and feedback [2].
- *Cloud-Native Integration:* Containerized AI agents can be embedded directly into CI/CD pipelines, enabling continuous and context-aware contributions throughout SDLC [25].

Current studies have explored various AI-assisted software development approaches, such as intelligent code completion (e.g., GitHub Copilot), requirement extraction from natural language documents, automated refactoring, and test case synthesis [30, 43]. While promising, these applications are typically phase-specific and do not exploit the full potential of AI autonomy.

Consequently, Agentic AI offers cross-phase integration, where agents maintain awareness of the entire project state and can transition seamlessly between phases. For example, an agent involved in requirements elicitation might assist in generating architectural models, then continue to validate these during coding and testing [6]. This continuous feedback loop is particularly aligned with modern agile and DevOps practices, where rapid iteration and adaptability are paramount.

However, despite growing interest in Agentic AI frameworks, systematic empirical evidence on their role in real-world SDLC contexts remains scarce. Most current literature focuses on proof-of-concept prototypes or isolated case studies in coding or testing automation [9, 20, 24, 27, 28]. There is limited understanding of how practitioners perceive Agentic AI's role, what challenges they anticipate, and what enabling factors could drive adoption across all phases of the SDLC [21, 22].

Therefore, addressing this knowledge gap requires qualitative, expert-driven research capable of capturing nuanced, practice-based insights from industry professionals who are actively engaged in integrating AI in software development. Thus, this study examines the insights of 21 industry experts, aiming to uncover the opportunities and challenges of integrating Agentic AI across all phases of the SDLC.

3 Motivation

The motivation for this study arises from two converging shifts: (i) accelerating delivery demands amid growing software complexity,

Table 1: Contrasting traditional AI, generative AI, and agentic AI in SE contexts.

Paradigm	Typical Capabilities	SE Implications
Traditional AI	Predictive models; rule-based automation	Phase-specific use; limited context awareness
Generative AI	Code/NL synthesis; summarization	Code completion, doc/test generation; mainly human-initiated
Agentic AI	Goal-directed planning; multi-step autonomy	Cross-phase continuity; proactive refactoring/testing; governance needs

and (ii) the emergence of Agentic AI as a potentially disruptive capability for automation, collaboration, and decision-making across the SDLC.

Over the last decade, faster digital transformation, adoption of cloud-native architectures, and integration of AI-driven business models have significantly shortened delivery timelines and raised quality expectations [21, 22]. Agile and DevOps have helped address these pressures through iterative delivery and automation [29]. However, they remain human-centric paradigms, where decision-making, coordination, and creativity are largely reliant on the development team's capacity. The increasing complexity of modern software systems driven by distributed architectures, integrated AI components, and global-scale deployments imposes substantial cognitive and coordination demands on development teams, often constraining scalability and operational efficiency [12].

In parallel, Agentic AI systems have emerged with capabilities far beyond conventional automation. Unlike static scripts or task-specific ML models, Agentic AI agents possess the ability to: *understand and reason about both natural language and source code, set subgoals and adapt strategies dynamically in response to changing project states, collaborate with humans and other agents to achieve shared objectives, retain long-term context across multiple SDLC phases.* These capabilities position Agentic AI as a *continuous, cross-phase contributor* in software engineering, rather than a niche tool for isolated tasks. For instance:

- In ***requirements engineering***, an Agentic AI could autonomously gather stakeholder feedback, reconcile conflicting requirements, and maintain a living requirements specification [44].
- In ***design***, it could simulate architectural trade-offs under different constraints, incorporating sustainability, performance, and maintainability considerations [35].
- In ***coding***, it could act as an adaptive pair programmer, capable of not only generating code but also explaining its decisions and aligning them with design documents[35, 41].
- In ***testing***, it could proactively generate test suites, execute them in CI/CD environments, and adjust based on defect trends [41].
- In ***maintenance***, it could monitor deployed systems, detect anomalies, suggest patches, and even coordinate with DevOps pipelines for deployment [23].

Despite these potentials, a realistic understanding of Agentic AI's integration into the SDLC is fragmented. Existing studies often focus narrowly on AI-powered code generation or bug detection without addressing: ***Cross-phase applicability*** — *how an AI agent's role in one phase influences and supports the next.* ***Organizational adoption challenges*** — *such as governance, trust, explainability, and integration into established workflows.* ***Human–AI collaboration patterns*** — *the evolving division of labor between developers, testers, product owners, and AI agents.*

This gap represents a timely opportunity to explore how Agentic AI can be operationalized as an end-to-end augmentation layer for the SDLC. From a practical perspective, industry stakeholders require evidence-based insights into *which SDLC phases benefit most from agentic capabilities, the maturity level of available tools and platforms, risks, constraints, and success factors for adoption.*

The motivation is therefore twofold: *(1) to extend the current understanding of AI in software engineering by mapping its potential and limitations when deployed as agentic systems across all SDLC phases, (2) to equip practitioners, technology leaders, and policymakers with actionable guidance on integrating Agentic AI into real-world development environments, ensuring that adoption delivers measurable gains in productivity, quality, and innovation capacity.* This study addresses these aims by conducting in-depth interviews with 21 domain experts, capturing both the opportunities and the challenges they foresee. The resulting analysis provides a phase-by-phase account of Agentic AI's role in software engineering, serving as a foundation for future empirical work and implementation frameworks.

4 Research Methodology

To explore the role of Agentic AI across the software development life cycle (SDLC), we adopted a qualitative research design grounded in semi-structured expert interviews [13]. This approach was chosen because the integration of Agentic AI in software engineering is still an emerging field, with few established metrics or standardized processes, making rich, context-dependent expert perspectives essential for understanding both opportunities and challenges.

4.1 Research Questions

We formulated the following research questions (RQs) to guide our study and structure the presentation of results:

RQ1: *Is Agentic AI suitable for the phases of the software development life cycle (SDLC)?*

Rationale: This question seeks to capture practitioners' perceptions of the significance and applicability of Agentic AI practices in software engineering. As Agentic AI is an emerging paradigm, it remains uncertain whether its practices can be directly applied or require adaptation to fit different SDLC phases.

RQ2: *What challenges hinder the adoption of Agentic AI in the SDLC?*

Rationale: This question aims to identify and classify the challenges that impede the adoption of Agentic AI practices in software engineering. Understanding these barriers can help highlight critical areas needing academic and industrial attention, thereby informing the development of guidelines and solutions to support effective and responsible adoption.

4.2 Expert Selection and Sampling

The study targeted software engineering professionals, AI researchers, and industry practitioners who have demonstrated experience with AI-assisted development tools, multi-agent systems, or intelligent automation in SDLC contexts. We used a purposive sampling strategy, supplemented by snowball sampling, to ensure diversity in expertise and roles. A total of 21 experts participated, drawn from: **(i) industry:** AI-powered software product companies, large IT service providers, and SMEs exploring AI adoption,**(ii) academia:** Universities and research institutes with active projects on AI in software engineering, agent-based systems, and human–AI collaboration, **(iii) hybrid roles:** Practitioners who bridge research and commercial deployment, including startup founders and innovation consultants. Expert selection aimed to cover the full SDLC spectrum, ensuring representation from: Requirements engineers and product managers, Software architects and designers, Developers and AI engineers, Test automation specialists, Maintenance and DevOps leads. This diverse sampling ensured that perspectives on Agentic AI integration were holistic, not restricted to a single phase of development.

4.3 Interview Design and Protocol

We conducted semi-structured interviews [13] to balance comparability across participants with the flexibility to explore unique insights. The interview guide was structured around the five key SDLC phases: requirements engineering, design, coding, testing, and maintenance, plus overarching themes such as cross-phase AI collaboration, adoption barriers, and organizational readiness. Example guiding questions included: *"How could Agentic AI support requirements elicitation, conflict resolution, and continuous validation?", "What role could AI agents play in architectural decision-making and trade-off analysis?", "How might Agentic AI change day-to-day development practices and pair programming?", "What capabilities could AI agents bring to automated test generation and execution?", "In what ways could Agentic AI support continuous monitoring, anomaly detection, and system evolution?", "How should organizations adapt workflows to fully leverage agentic capabilities?"*

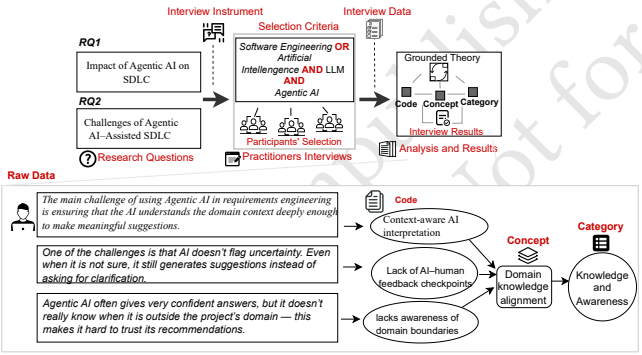

Figure 1: Used research method

Each interview lasted 30–40 minutes and was conducted via Zoom, Microsoft Teams, or in-person where feasible. With participant consent, all sessions were audio-recorded and later transcribed verbatim for analysis.

4.4 Data Collection

Data collection was carried out over a four-month period using an iterative and reflexive process. Participants were initially recruited through professional networks, ongoing research collaborations, and targeted LinkedIn invitations to ensure diversity across roles, domains, and geographical contexts. To refine the interview guide and confirm the clarity of the questions, two pilot interviews were conducted, after which minor adjustments were made to the sequencing and wording of prompts. The main interview round followed, where semi-structured interviews allowed for both comparability across participants and flexibility in exploring emerging insights. As new themes surfaced, the researchers refined their probing questions to capture rich, practice-oriented experiences beyond the predefined scope. To enhance the credibility and trustworthiness of the findings, participants were subsequently invited to review selected excerpts and preliminary interpretations in a post-interview validation step, ensuring that the researchers' coding and thematic framing accurately reflected the experts' intended meanings. This iterative approach balanced structure with openness, allowing the study to capture both shared patterns and context-specific nuances of Agentic AI adoption in software engineering.

4.5 Data Analysis

We employed a grounded theory approach for analyzing the data [16], using NVivo15l for coding and data management. The analysis began with a phase of familiarization, during which each transcript was read multiple times to capture initial impressions and recurring ideas. This was followed by open coding, where preliminary codes were generated to represent relevant concepts such as "AI for conflict resolution in requirements," "autonomous test case generation," and "AI explainability concerns." In the next stage, axial coding was used to group related codes into broader categories that aligned with the different phases of the software development life cycle. Building on this, selective coding was applied to identify cross-phase themes, including recurring patterns such as "human–AI collaboration" and "trust and accountability mechanisms." To strengthen the credibility of the analysis, synthesized results were shared with a subset of participants for expert validation, ensuring that interpretations accurately reflected their perspectives. Through this iterative process, codes were ultimately organized into both phase-specific and cross-phase themes, allowing the results to be presented sequentially for each SDLC phase as well as holistically to highlight broader patterns across the entire development cycle. Based on the semi-structured interview questions described in the above section, the analysis of collected data followed a three-step process.

Step 1 – Demographic profiling of participants

The first step focused on analyzing the demographic details of the 21 experts who participated in the study, including their country of residence, years of professional experience, primary domain of expertise, and role within the software development life cycle. These demographic insights are given in Figure 2 and provide an

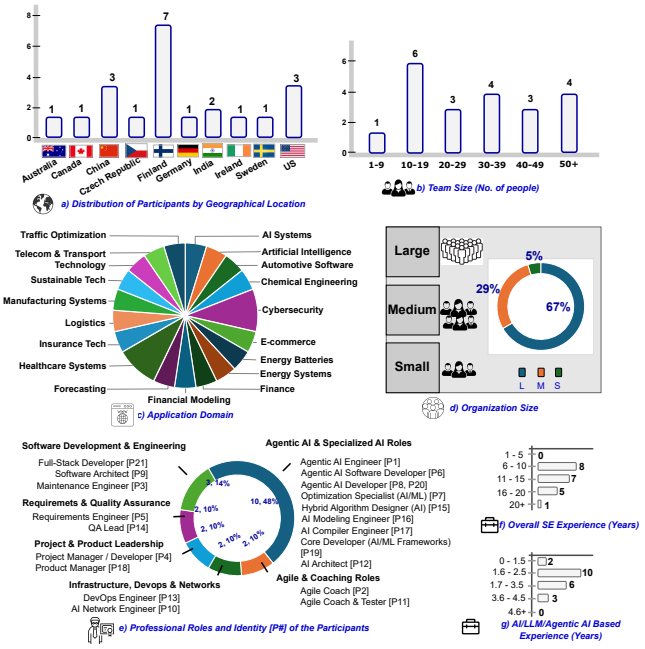

Figure 2: Demographic of interview participants

important and contextual interpretation for the results of RQ1 and RQ2. For example, examining whether variables such as experience length, professional domain *(e.g., Automotive Software, Energy Systems, Finance, Insurance Tech, Logistics, Manufacturing Systems), or role (e.g. Agentic AI Engineer, Agile Coach, Maintenance Engineer, Project Manager / Developer, Requirements Engineer, Agentic AI Software Engineer, Optimization Specialist (AI/ML))* influenced participants' perspectives on the integration of Agentic AI in SDLC phases.

Step 2 – Structured ratings and qualitative elaboration

In the second step, data for RQ1 were collected through a 5-point Likert scale *(Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree)*, followed by an open-ended justification to capture the reasoning behind each rating (see Figure 3, section 5.1). For example, when participants rated the potential of Agentic AI to reduce human error in requirements validation, they were prompted to elaborate on specific experiences, perceived limitations, or situational dependencies influencing their view.

Step 3 – Grounded Theory analysis for deeper insights

The third step, focusing on RQ2, applied the Grounded Theory (GT) method [16] to systematically derive codes, concepts, and categories (Figure 1) from the qualitative data. GT was chosen for multiple reasons:

- The application of Agentic AI to the entire SDLC is a novel, underexplored research area, similar to how GT has been successfully used in software engineering for emerging domains [18, 26].
- The methodology's emphasis on social and collaborative processes aligns with studying AI–human team interactions, decision-making, and adaptive workflows in software development.

We followed open coding [5] and constant comparison [16] to iteratively process raw transcripts into higher levels of abstraction. The initial coding was conducted by the first two authors, with the remaining authors participating in the final coding review to ensure interpretive accuracy.

Example of the coding process.

Raw data: one of the practitioners stated that 💬 *"The main challenge of using Agentic AI in requirements engineering is ensuring that the AI understands the domain context deeply enough to make meaningful suggestions."* [P5, Requirements Engineer]

Code: *Context-aware AI interpretation:* This code was found to be conceptually similar to others, as stated by practitioners: 💬 *"AI's inability to capture implicit stakeholder needs"* [P07, Product Manager] and *"Need for domain-specific AI training data"* [P20, Agentic AI Developer].

Concept: *Domain knowledge alignment:* Other concepts emerging from similar comparisons included *"human–AI trust calibration"* and *"continuous stakeholder feedback loops"*. Through final constant comparison, these related concepts were grouped into a higher-order Category: Knowledge and Awareness.

Categories: *The Knowledge and Awareness:* category captures participants' concerns regarding the understanding, contextual reasoning, and explainability of Agentic AI systems across SDLC phases. This category reflects the emphasis participants placed on the need for domain-adapted AI models, accessible training resources, and trust-building mechanisms for effective human–AI collaboration. Other high-level categories emerging from the data included:

- **Sustainable Integration** – Ensuring that Agentic AI adoption strategies remain adaptable to evolving tools, standards, and project contexts.
- **Tooling and Automation Ecosystem** – Availability of integrated agentic development environments, testing suites, and deployment pipelines.
- **Governance and Ethical Compliance** – Standards, policies, and safeguards ensuring responsible AI use in software engineering.

The abstraction levels of ***codes → concepts → categories*** in this study are firmly grounded in real-world practitioner experiences. While we do not claim universal generalizability, the categories accurately represent the current state of practitioner knowledge and expectations regarding Agentic AI in the SDLC. The categories are explored in detail in the section 5, with direct participant quotations supporting the thematic analysis.

5 Results

This section presents the study findings in response to the two research questions outlined in Section 4. The demographic analysis contextualizes the results by examining the diversity and experience levels of the targeted population, enabling a better understanding of their perspectives on Agentic AI in software engineering (Figure 2). A total of 21 experts from 10 countries across 4 continents participated in this study, representing 20 distinct professional roles and 19 industrial domains (Figure 2a, 2c, 2e). Participants' experience in AI-assisted software development ranged from 1 to 20+ years, with a significant portion (n=15) having between 6–15 years of overall

software engineering experience (Figure 2f). In terms of Agentic AI experience specifically, most participants (n=18) reported 0–3.5 years, reflecting the novelty of the field (Figure 3g). The participant pool included software architects, requirements engineers, AI engineers, DevOps specialists, agile coaches, data scientists, and product managers, among others, allowing for multiple viewpoints across SDLC phases. This diversity across geographies, roles, and domains lends credibility to the results and increases confidence in applying these findings more broadly, though not universally. The following sections address RQ1 (practitioner perspectives on Agentic AI applicability in SDLC) and RQ2 (key challenges and enablers of Agentic AI in SDLC), with detailed thematic breakdowns for each SDLC phase.

5.1 Impact of Agentic AI on SDLC (RQ1)

This section presents the experts perceptions regarding the role of Agentic AI across SDLC phases.

5.1.1 Agentic AI in Requirements Engineering. To assess perceptions of Agentic AI's potential in requirements elicitation, analysis, and validation, participants rated applicability on a five-point Likert scale (Figure 3), followed by open-ended rationales. A total of 12 participants strongly agreed and 4 agreed (n=16) that Agentic AI can significantly enhance requirements engineering, particularly by *(i) automating stakeholder communication analysis, (ii) detecting ambiguities and contradictions in requirement statements, and (iii) recommending domain-aligned user stories.* As one of the practitioner stated that: *"With Agentic AI, we can automatically detect gaps between verbal stakeholder discussions and written requirements, something that usually costs weeks of manual reviews."* [P05, Requirements Engineer]

However, 4 participants were neutral and 1 disagreed, citing concerns about context comprehension, bias in training data, and over-reliance on automated outputs. As one practitioner noted: *"AI can draft requirements, but without deep domain understanding, it risks introducing false assumptions that slip past human reviewers."* [P18, Product Manager]

Neutral participants highlighted that while AI could be useful for drafting and validation, *"it lacks the tacit knowledge and stakeholder empathy needed for prioritizing requirements in fast-changing domains."* [P07, Product Owner]

Knowledge and Awareness – Requirements engineers emphasized the need for domain-adapted LLMs, training datasets that reflect industry-specific terminology, and explainable reasoning mechanisms to build trust.

5.1.2 Agentic AI in Software Design. Most participants (n=17) agreed that Agentic AI is a valuable co-pilot in architectural modelling, design pattern selection, and system decomposition, especially for rapid prototyping. They highlighted advantages such as automated UML diagram generation, dependency analysis, and simulation of design trade-offs. One of the practitioner stated that: *"Agentic AI can evaluate multiple architecture blueprints in parallel, flagging bottlenecks before we even start implementation."* [P09, Software Architect]

However, 3 participants were neutral and 1 disagreed. Neutral participants felt that AI-assisted modelling was promising but *"often over-focused on performance optimization and missed maintainability or security trade-offs that human architects must still balance."* [P12, Agile Coach] The disagreeing participant emphasized: *"AI tools propose designs without fully grasping long-term business goals, which makes them risky for mission-critical systems."* [P02, Senior Architect]

Tooling and Automation Ecosystem – Practitioners called for integrated agentic design assistants that connect seamlessly with modelling tools like Enterprise Architect, ArchiMate, and cloud architecture blueprints.

5.1.3 Agentic AI in Software Coding. All participants (n=21) strongly agreed or agreed that Agentic AI could accelerate coding through: *(i) auto-generation of function scaffolds and boilerplate code, (ii) automated refactoring recommendations, (iii) code style enforcement aligned with project standards.* As one of the practitioners stated that: *For instance, one of the participants mentioned that "My team used Agentic AI to refactor a 20K-line codebase for cloud migration—it suggested optimisations I wouldn't have considered."* [P21, Full-Stack Developer]

Nonetheless, even within agreement, concerns were voiced as stated by the practitioner: *"The generated snippets sometimes introduced subtle security flaws that were not immediately visible."*[P04, Project Manager / Developer]. Similarly, another practitioner warned that *"AI-generated code could mask technical debt if version control doesn't explicitly track its origin."* [P19, Core Developer (AI/ML Frameworks)]

Sustainable Integration – Developers stressed the need for continuous auditing pipelines to validate AI-generated code and version control mechanisms that track AI contributions for accountability.

5.1.4 Agentic AI in Software Testing. Most respondents (n=18) agreed that Agentic AI could automate test case generation, mutation testing, and coverage analysis. They saw value in AI's ability to: *(i) derive tests directly from requirements or code, (ii) simulate user behavior patterns for usability testing, (iii) predict defect-prone areas before release.* As one of the practitioners stated: *"Agentic AI can execute exploratory testing in ways humans can't, uncovering hidden states in complex systems."* [P14, QA Lead]

However, 2 participants were neutral and 1 disagreed. Neutral participants reasoned that *"AI-generated tests often miss industry-specific edge cases and need substantial human oversight to be production-ready."* [P08, Test Automation Specialist]. The disagreeing participant argued: *"False positives waste as much time as they save—without domain calibration, I don't trust the results enough to integrate them directly."* [P11, QA Engineer]

Governance and Ethical Compliance – Test engineers emphasized AI transparency in test result explanations and traceability from requirement → test → defect report.

5.1.5 Agentic AI in Software Maintenance. A strong majority (n=18) recognized Agentic AI's role in predictive maintenance, legacy code comprehension, and automated documentation updates. AI-powered dependency scanning was also seen as a proactive way to manage security vulnerabilities, as stated by one of the practitioners: *"For a legacy ERP system, Agentic AI mapped undocumented*

modules to business functions in hours—something that would take weeks manually." [P03, Maintenance Engineer]

However, 3 participants remained neutral, reflecting reservations. They noted that *"AI's reliance on production logs poses privacy and compliance risks, especially in regulated industries."* [P10, DevOps Lead]. Another neutral participant added: *"Model drift makes it unreliable for long-term monitoring unless retraining pipelines are strictly controlled."* [P06, Maintenance Engineer]

Continuous SE Infrastructure – Participants proposed AI-powered observability dashboards that connect operational monitoring with code-level insights, enabling real-time adaptive maintenance.

5.2 Challenges of Agentic AI–Assisted SDLC (RQ2)

Agentic AI in software engineering introduces a broad range of challenges across people-centric, technology-centric, process-centric, and governance-centric dimensions (Figure 3).

5.2.1 Agentic AI awareness and skills gap. Participants frequently noted a lack of *Agentic AI literacy* within development teams — i.e., limited understanding of autonomous reasoning frameworks, prompt engineering, model orchestration, and continuous AI feedback loops. One of the practitioners stated: *"It's not enough to know AI APIs. You need people who understand how autonomous agents negotiate, delegate, and resolve conflicts — otherwise the whole system becomes unpredictable."* [P12, AI Architect]

This lack of expertise suggests an urgent need for formal education and structured training pathways, enabling developers to extend existing software engineering skillsets with AI orchestration, ethics, and multi-agent system design.

5.2.2 Ethically aligned AI–human ecosystems. Beyond technical integration, participants emphasized the socio-technical risks of delegating decision-making to autonomous agents. Ethical implications such as bias amplification, data misuse, and opaque decision chains demand robust governance models. As stated by one of the practitioners: *"Agentic AI changes accountability. If an autonomous agent makes a wrong call in requirements prioritization, who's responsible — the developer, the product owner, or the AI provider?"* [P18, Product Manager]

Therefore, developing an AI–human ecosystem that embeds ethics-by-design, domain-specific compliance checks, and traceable decision logs was seen as critical.

5.2.3 Tooling maturity and continuous integration. Similar to the quantum software case, participants identified a shortage of agent-aware development tools that integrate seamlessly into CI/CD pipelines. Current AI coding assistants and testing tools often operate in isolation, lacking context persistence across SDLC stages. As one of the practitioners stated: *"We need AI agents that not only generate code but also feed lessons back into design, test, and deployment phases — right now, most tools don't talk to each other."* [P13, DevOps Engineer]

5.2.4 Standards and minimal-but-sufficient documentation. While agile principles discourage excessive documentation, participants acknowledged that Agentic AI workflows require clear governance artefacts such as operational boundaries for agents, explainability protocols, and fallback procedures. As one of the practitioners stated: *"You don't need 500-page specs, but you do need enough shared documentation so that humans and AI agents can work from the same mental model."* [P2, Agile Coach]

6 Discussion

We now summarize the study's core findings (RQ1, RQ2), discuss their implications for research and practice, and present potential threats to validity.

6.1 Agentic AI in the Software Development Life Cycle (RQ1) and Related Challenges (RQ2)

Agentic AI can support software teams by handling tasks that are repetitive, knowledge-heavy, or time-consuming. Instead of humans managing every detail manually, AI agents can work alongside developers in an iterative and incremental way. For example, an AI system can help translate business goals into design models, generate architecture options, create code scaffolds, and test them against stakeholder feedback. In this way, AI reduces manual overhead while allowing developers to focus on higher-level decisions.

Most participants (18 out of 21) agreed that Agentic AI can make the software development process faster and more effective. They saw it as a natural next step in software engineering, where AI tools are integrated into everyday workflows to reduce effort in requirements gathering, design, coding, testing, and maintenance. Participants pointed to several specific benefits, such as using AI to convert requirements into design models, refining outputs across different phases of the SDLC, and supporting human–AI co-design, where humans and AI share decision-making responsibility.

However, not all views were positive. Ten participants raised concerns about the immaturity of current tools, the lack of domain-specific training data, and gaps in expertise that make it difficult to integrate AI smoothly into existing pipelines. Six participants were neutral, saying that successful adoption depends heavily on organizational readiness, governance, and explainability. These perspectives show that while the potential is high, adoption challenges are also very real.

As shown in Figure 3, participants' responses (agree, neutral, disagree) were mapped to each phase of the SDLC and then grouped into four broad categories: *Knowledge and Awareness, Tooling and Automation Ecosystem, Sustainable Integration,* and *Governance and Ethical Compliance.* This mapping shows how Agentic AI can be both a powerful enabler (RQ1) and a source of challenges (RQ2).

On the positive side, participants highlighted clear advantages in every phase of the SDLC. In the requirements phase, AI can spot missing or conflicting requirements and suggest user stories. During design, it can speed up architecture modelling and evaluate design trade-offs. In coding, AI helps by generating scaffolds, suggesting refactors, and enforcing style rules. For testing, AI is able to generate test cases, predict defects, and simulate user behaviour. Finally, in maintenance, AI supports predictive monitoring, keeps documentation current, and scans for vulnerabilities.

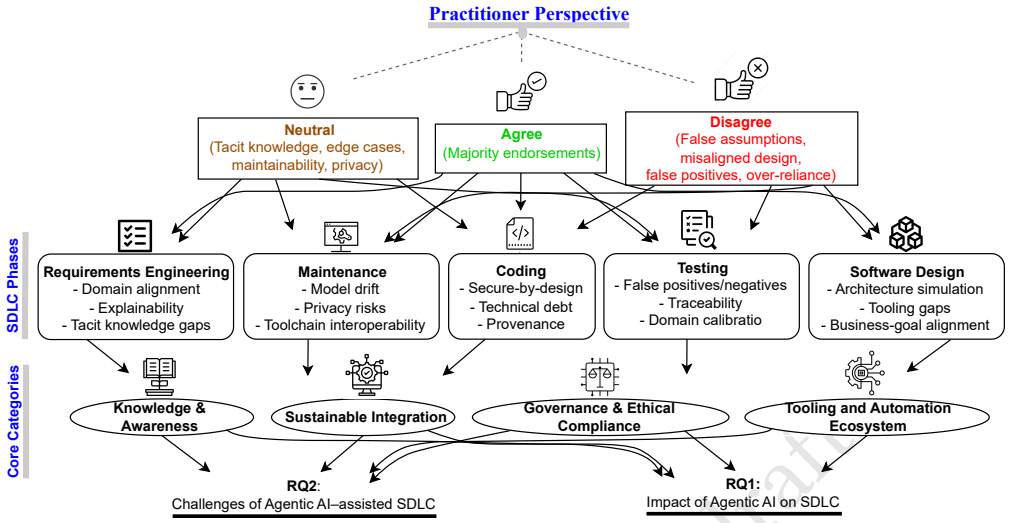

Figure 3: A holistic view of practitioners' opinions.

At the same time, neutral and disagreeing participants pointed out key risks that must be addressed. Under the category of *Knowledge and Awareness*, participants described an **Agentic AI awareness and skills gap**, noting that many teams lacked literacy in autonomous reasoning frameworks, prompt engineering, and multi-agent orchestration. As one architect explained: 💬 *"It's not enough to know AI APIs. You need people who understand how autonomous agents negotiate, delegate, and resolve conflicts — otherwise the whole system becomes unpredictable."* [P12, AI Architect]. This points to an urgent need for training and structured pathways that combine technical AI knowledge with ethics and system design.

Within *Governance and Ethical Compliance*, practitioners raised the issue of **ethically aligned AI–human ecosystems**. They were concerned about accountability, bias, and opaque decision-making. As one product manager stressed: 💬 *"Agentic AI changes accountability. If an autonomous agent makes a wrong call in requirements prioritization, who's responsible — the developer, the product owner, or the AI provider?"* [P4, Product Manager]. This highlights the importance of embedding ethics-by-design, domain-specific compliance checks, and traceable decision logs.

From the perspective of the *Tooling and Automation Ecosystem*, participants identified **tooling maturity and continuous integration** as major gaps. Existing AI tools often operate in isolation, without persistence of context across SDLC stages. As a DevOps engineer put it: 💬 *"We need AI agents that not only generate code but also feed lessons back into design, test, and deployment phases — right now, most tools don't talk to each other."* [P13, DevOps Engineer]. This suggests that real-world integration of agentic tools into CI/CD pipelines remains immature.

Finally, under *Sustainable Integration*, participants emphasized the need for **standards and minimal-but-sufficient documentation**. While agile methods discourage excessive documentation, AI-driven workflows require clear boundaries, protocols for explainability, and fallback mechanisms. As one agile coach noted: 💬 *"You don't need 500-page specs, but you do need enough shared documentation so that humans and AI agents can work from the same mental model."* [P2, Agile Coach]. Without such governance artefacts, sustainable adoption will be difficult.

To summarize, Figure 3. gives a holistic picture of practitioners' views. Agentic AI is widely seen as a promising enabler that can reduce effort and improve quality across SDLC phases. At the same time, it introduces challenges that relate not just to technology but also to people, processes, and governance. For successful adoption, organizations will need to combine technical readiness with strong governance frameworks, domain-specific training, and effective human–AI collaboration.

6.2 Implications for Research and Practice

The findings of this research entail several implications for research and practice, discussed as follows:

Implications for research:– This study extends ongoing work in AI-assisted software engineering by offering practitioner-based, empirically grounded categories of both enablers and challenges. Researchers can build on these findings to test new ideas about how tasks should be divided between humans and AI agents, and under what conditions such division improves performance and trust. Another key area for future work is governance: there is a strong need to design and evaluate frameworks that ensure accountability, transparency, and ethical compliance in agent–human collaboration. Finally, the results point to the importance of toolchain integration. Research should focus on creating and validating patterns that allow AI support to persist across the whole development lifecycle, instead of remaining isolated in single phases.

Implications for practice:- For practitioners, the study provides concrete guidance on how to start operationalizing Agentic AI in real-world projects. The categories of enablers and challenges can be used to shape roadmaps for AI-driven transformation of SDLC pipelines. At the organizational level, companies should invest in skill development programs that focus not only on the technical orchestration of AI tools but also on ethics and responsible use.

When selecting or configuring AI tools, decision-makers should prioritize those that offer interoperability and support continuous learning, so that improvements can accumulate across projects. In practice, this means that successful adoption requires more than just adopting new tools — it requires building the right mix of technical capacity, governance processes, and cultural readiness to support sustainable integration of Agentic AI. One of the AI tools that supports interoperability and continuous learning.

6.3 Threats to Validity

The findings of this study are subject to several potential threats to validity.

Internal validity:– Risks include participant misinterpretation of interview questions and self-reporting bias. We mitigated these by piloting the interview guide, selecting only participants with both software engineering and AI project experience, and conducting regular calibration meetings among the research team. ***Construct validity:-*** Core constructs (e.g., "Agentic AI applicability", "challenge categories") were defined iteratively and validated through multiple coding cycles, with direct participant quotes used to ground interpretations. ***External validity:-*** While participants represented 10 countries and 20 roles, the field of Agentic AI–assisted SDLC is still nascent, limiting generalizability. Broader validation will require larger-scale studies, including industrial case studies and quantitative surveys. ***Conclusion validity:-*** The research team engaged in continuous peer review of data interpretation, with multiple authors cross-checking thematic assignments. Disagreements were resolved through consensus during joint analysis sessions.

7 Related Work

We discuss the most relevant existing work, reviewing the state-of-the-art on processes and development of lifecycles for AI-assisted software systems, with a particular focus on Agentic AI. A conclusive summary at the end reiterates the scope and contributions of the proposed research.

7.1 Process-Centred Engineering of Agentic AI–Assisted Software

Agentic AI–assisted software engineering is an emerging paradigm that aims to embed autonomous AI agents into existing software processes, practices, methods, tools, and techniques to support the design, development, testing, and maintenance of software systems more effectively and efficiently. While the existing body of software engineering knowledge provides strong foundations for AI-assisted development [17], the unique characteristics of Agentic AI such as autonomous decision-making, multi-agent orchestration, and adaptive role delegation require traditional methods to be tailored to address AI-specific challenges.

Several recent systematic reviews and conceptual studies have explored the integration of AI into the software lifecycle, focusing on aspects such as requirements automation [37], agent-supported architectural design [45], automated code generation and review [7], and continuous AI-driven testing and deployment [40]. These efforts aim to establish empirical foundations for AI-assisted software engineering by streamlining processes, identifying reusable patterns, and building toolchains that empower traditional software engineers to act as AI orchestrators rather than only code implementers.

While the field is still in its early stages, community-wide initiatives are growing rapidly. Dedicated workshops (e.g., AI-SE), conferences on AI-assisted programming, and repositories mining for AI-in-code contexts [7, 37] demonstrate a shared emphasis on capturing best practices, reference architectures, and reusable knowledge for integrating Agentic AI into the SDLC.

7.2 Development in Agentic AI–Assisted Software Engineering

To align with iterative and incremental software development, several studies have presented visions for continuous, AI-assisted development and delivery [37]. In such paradigms, AI agents act not as static tools but as persistent collaborators learning from past iterations, adapting to evolving requirements, and proactively suggesting improvements across SDLC phases.

Despite conceptual models being proposed for AI-assisted iteration, empirical maturity remains low. Existing methods [9, 20, 24, 27, 28, 37, 40] reveal that there is no consensus on the best practices for integrating agentic autonomy into iterative processes. However, early work by Sapkota et al. [38] proposed a conceptual taxonomy, applications and challenges of AI-augmented workflows, enabling adaptive task delegation between humans and AI agents. This taxonomy, while conceptually sound, lacks empirical evaluation in large-scale, real-world projects.

Similarly, White [42] introduced building living software systems with generative & agentic AI that mirrors traditional lifecycle phases but integrates agent-specific orchestration stages, such as autonomous backlog prioritization, cross-phase context tracking, and AI–human negotiation protocols. These AI-specific stages enable the distinction between activities that can be fully automated by agents and those requiring human oversight, akin to the "quantum-classical split" proposed in quantum software engineering but adapted for human–AI collaboration.

7.3 Conclusive Summary

In this study, it is argued that before adopting any specific processes or reference models for Agentic AI–assisted software engineering, there is a need to analyze practitioner perceptions, professional practices, and empirically grounded insights on the potential and challenges of integrating autonomous AI agents across the SDLC.

Our work complements existing efforts that focus on frameworks [34], architecture [35], and processes [45] for AI-assisted software by contributing interview-based, practitioner-centric evidence. This approach bridges conceptual visions with real-world constraints, helping to systematize the integration of Agentic AI in all SDLC phases — from requirements engineering to maintenance, while identifying the key enablers, barriers, and governance requirements for successful adoption.

8 Conclusions and Future Work

Agentic AI–assisted software engineering extends the principles of classical software engineering by enabling architects, developers, and operations teams to collaborate with autonomous AI agents

in delivering adaptive, high-quality software applications. This paradigm offers new momentum for digital transformation across industries while redefining the boundaries of human–machine collaboration.

Grounded in empirical analysis and thematic coding, this study examined practitioners' perspectives on the opportunities and pitfalls of integrating Agentic AI across all phases of the software development life cycle. We engaged 21 practitioners from 10 countries, spanning roles such as software engineers, AI/ML specialists, agile experts, and research engineers (see Figure 3). Participants widely agreed that Agentic AI could substantially enhance software development efficiency and innovation, although some expressed caution regarding its maturity and suitability for all phases. This addresses RQ1 by showing both strong optimism and measured skepticism within the professional community.

To understand these nuances, we investigated perceived challenges (RQ2) through open-ended discussions and grounded theory analysis. Four major challenge categories emerged:

(1) ***Knowledge and Awareness*** – skill gaps and conceptual disconnects between traditional SE roles and the operational mindset required for AI–agent collaboration, including limited interpretability of agent reasoning and insufficient guidelines for AI–human interaction.
(2) ***Sustainable Scaling*** – concerns about long-term maintainability, ethical alignment, and governance, with risks to transparency, accountability, and trust when agents make critical development decisions without oversight.
(3) ***Agent-Aware Tools and Infrastructure*** – immaturity of current tools to integrate, configure, and monitor AI agents within CI/CD pipelines, and the lack of standardized interfaces for multi-agent orchestration.
(4) ***Standards and Specifications*** – the absence of universally accepted benchmarks, maturity models, and documentation practices that can guide safe and consistent adoption of Agentic AI in the SDLC.

From the practitioners' perspective, successful adoption of Agentic AI requires not only refining existing SE practices but also co-creating agent-oriented process models, governance frameworks, and educational resources.

8.1 Future Work

Building on these insights, future research should identify additional challenges and enablers through large-scale empirical evidence. In particular, we plan to:

(1) Mine open-source repositories and developer forums (e.g., GitHub, GitLab, Stack Overflow) to capture real-world adoption patterns, best practices, and recurring issues in agent-assisted development.
(2) Map these observed practices to the challenge categories identified in this study, linking technical interventions with governance models and socio-technical adoption factors.
(3) Validate the resulting framework through a large-scale quantitative survey, complemented by follow-up interviews across diverse industry sectors, to assess generalizability.

By combining empirical insights with community-driven evidence, our long-term goal is to create a robust, practitioner-validated framework for Agentic AI–assisted SDLC. Such a framework should balance automation with adaptability, embed ethical responsibility into workflows, and ultimately help organizations unlock the full potential of autonomous AI agents in software engineering.

9 Acknowledgments

The research team would like to thank the interview participants and the industrial partner who generously participated and helped us in this research, and shared their time and experience.

References

[1] Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. 2025. Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. *IEEe Access* (2025).
[2] Bolaji Iyanu Adekunle, Ezinne C Chukwuma-Eke, Emmanuel Damilare Balogun, and Kolade Olusola Ogunsola. 2021. Machine learning for automation: Developing data-driven solutions for process optimization and accuracy improvement. *Machine Learning* 2, 1 (2021).
[3] Tanvir Ahmed, Samiul Hasan, Ahammed Shorif, Ansarul Hoque, Shadman Sajid, and Md Badiuzzaman Biplob. 2025. Secure Engineering of Autonomous AI Agents: A Threat-Driven Development Framework. (2025).
[4] Daniel Ajiga, Patrick Azuka Okeleke, Samuel Olaoluwa Folorunsho, and Chinedu Ezeigweneme. 2024. Enhancing software development practices with AI insights in high-tech companies. *IEEE Software Engineering Institute, Technical Report TR-2024-003* (2024).
[5] George Allan. 2003. A Critique of Using Grounded Theory as a Research Method. *Electronic Journal of Business Research Methods* 2, 1 (2003), 1–10.
[6] Mohammadmehdi Ataei, Hyunmin Cheong, Daniele Grandi, Ye Wang, Nigel Morris, and Alexander Tessier. 2025. Elicitron: A large language model agent-based simulation framework for design requirements elicitation. *Journal of Computing and Information Science in Engineering* 25, 2 (2025), 021012.
[7] Zouhair Ibn Batouta, Rachid Dehbi, Mohammed Talea, and Omar Hajoui. 2016. Automation in code generation: Tertiary and systematic mapping review. In *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*. IEEE, 200–205.
[8] Luca Berardinelli, Vittoriano Muttillo, Romina Eramo, Hugo Bruneliere, Abbas Rahimi, Antonio Cicchetti, Joan Giner-Miguelez, Abel Gomez, Pasqualina Potena, and Mehrdad Saadatmand. 2025. Model Driven Engineering, Artificial Intelligence, and DevOps for Software and Systems Engineering: A Systematic Mapping Study of Synergies and Challenges. *ACM Transactions on Software Engineering and Methodology* (2025).
[9] Luigi Colucci Cante and Antonio Esposito. 2025. Artificial Intelligent Technologies to Support Software Engineering: A Survey. In *Complex, Intelligent and Software Intensive Systems: Proceedings of the 19th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2025), Volume 2*, Vol. 261. Springer Nature, 240.
[10] Nadia Chafik and Dr Amine Benchekroun. 2020. Integrating Artificial Intelligence in Software Engineering: Enhancements and Challenges in the Development Lifecycle. (2020).
[11] Hamza Chniter, Yuting Li, Mohamed Khalgui, Anis Koubaa, Zhiwu Li, and Fethi Jarray. 2018. Multi-agent adaptive architecture for flexible distributed real-time systems. *IEEE Access* 6 (2018), 23152–23171.
[12] Vaishali Dhanoa, Anton Wolter, Gabriela Molina León, Hans-Jörg Schulz, and Niklas Elmqvist. 2025. Agentic Visualization: Extracting Agent-based Design Patterns from Visualization Systems. *arXiv preprint arXiv:2505.19101* (2025).
[13] Patric Finkbeiner. 2016. Qualitative research: semi-structured expert interview. In *Social Media for Knowledge Sharing in Automotive Repair*. Springer, 141–181.
[14] Venus Garg. 2025. Designing the Mind: How Agentic Frameworks Are Shaping the Future of AI Behavior. *Journal of Computer Science and Technology Studies* 7, 5 (2025), 182–193.
[15] Rafi Giffari, M Mushlih Ridho, Dana Indra Sensuse, Deden Sumirat Hidayat, and Erisva Hakiki Purwaningsih. 2024. Analyst's Perception on the Use of AI-based Tools in the Software Development Life Cycle. *Jurnal Sistem Informasi* 20, 1 (2024), 73–87.
[16] Barney Glaser and Anselm Strauss. 2017. *Discovery of grounded theory: Strategies for qualitative research*. Routledge.
[17] Johan Gottlander and Theodor Khademi. 2023. The Effects of AI Assisted Programming in Software Engineering. (2023).
[18] Rashina Hoda, James Noble, and Stuart Marshall. 2010. Organizing self-organizing teams. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. 285–294.
[19] Mohammad Ikbal Hossain. 2023. Software development life cycle (SDLC) methodologies for information systems project management. *International Journal for*

Multidisciplinary Research 5, 5 (2023), 1–36.
[20] Ken Huang. 2025. *Agentic AI.* Springer.
[21] Laurie Hughes, Yogesh K Dwivedi, Keyao Li, Mandanna Appanderanda, Mousa Ahmad Al-Bashrawi, and Inyoung Chae. 2025. AI Agents and Agentic Systems: Redefining Global it Management. 11 pages.
[22] Laurie Hughes, Yogesh K Dwivedi, Tegwen Malik, Mazen Shawosh, Mousa Ahmed Albashrawi, Il Jeon, Vincent Dutot, Mandanna Appanderanda, Tom Crick, Rahul De', et al. 2025. AI agents and agentic systems: A multi-expert analysis. *Journal of Computer Information Systems* (2025), 1–29.
[23] Wenyu Jiang and Fuwen Hu. 2025. Artificial Intelligence Agent-Enabled Predictive Maintenance: Conceptual Proposal and Basic Framework. *Computers* 14, 8 (2025), 329.
[24] Satyadhar Joshi. 2025. LLMOps, AgentOps, and MLOps for Generative AI: A Comprehensive Review. (2025).
[25] R Karthick. 2025. A Comprehensive Survey on AI-Enabled Cloud Security, DevSecOps, and Scalable Digital Infrastructure. (2025).
[26] Kashumi Madampe, Rashina Hoda, and John Grundy. 2021. A faceted taxonomy of requirements changes in agile contexts. *IEEE Transactions on Software Engineering* 48, 10 (2021), 3737–3752.
[27] Marvee Mahmood. 2025. The Configurations of Human Expertise with AI across SoftwareDevelopment Lifecycle.
[28] Nikhil Sagar Miriyala, Bharath Kumar Bandaru, Prakhar Mittal, Kiran Babu Macha, Rishi Venkat, and Anu Rai. [n. d.]. An Efficient Solution towards SDLC Automation using Multi-Agent Integration through Crew AI. ([n. d.]).
[29] Akshay Mittal. 2025. AI-Driven DevOps Automation for Cloud-Native Application Modernization. *Authorea Preprints* (2025).
[30] Nouman Noor. 2025. Generative AI-assisted software development teams: opportunities, challenges, and best practices. (2025).
[31] Ihor Pysmennyi, Roman Kyslyi, and Kyrylo Kleshch. 2025. AI-driven tools in modern software quality assurance: an assessment of benefits, challenges, and future directions. *arXiv preprint arXiv:2506.16586* (2025).
[32] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924* (2023).
[33] Partha Pratim Ray. 2025. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *Authorea Preprints* (2025).
[34] Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468* (2025).
[35] Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Vibe coding vs. agentic coding: Fundamentals and practical implications of agentic ai. *arXiv preprint arXiv:2505.19443* (2025).
[36] Jaakko Sauvola, Sasu Tarkoma, Mika Klemettinen, Jukka Riekki, and David Doermann. 2024. Future of software development with generative AI. *Automated Software Engineering* 31, 1 (2024), 26.
[37] Arpita Soni, Anoop Kumar, Rajeev Arora, and Ramakrishna Garine. 2023. Integrating AI into the Software Development Life Cycle: Best Practices, Tools, and Impact Analysis. *Tools, and Impact Analysis (June 10, 2023)* (2023).
[38] Petteri Teikari, Mike Jarrell, Maryam Azh, and Harri Pesola. 2025. The Architecture of Trust: A Framework for AI-Augmented Real Estate Valuation in the Era of Structured Data. *arXiv preprint arXiv:2508.02765* (2025).
[39] Akkala Teja Swaroop. 2025. The Transformative Impact Of Artificial Intelligence On Professional Software Development: A Comprehensive Analysis. *Title of Paper: The Transformative Impact Of Artificial Intelligence On Professional Software Development: A Comprehensive Analysis published in Page No* 13, 8 (2025), b74–b90.
[40] Bharath Chandra Vadde and VB Munagandla. 2023. Integrating AI-Driven Continuous Testing in DevOps for Enhanced Software Quality. *Revista de Inteligencia Artificial en Medicina* 14, 1 (2023), 505–513.
[41] Huanting Wang, Jingzhi Gong, Huawei Zhang, and Zheng Wang. 2025. AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities. *arXiv preprint arXiv:2508.11126* (2025).
[42] Jules White. 2024. Building living software systems with generative & agentic AI. *arXiv preprint arXiv:2408.01768* (2024).
[43] Man-Fai Wong, Shangxin Guo, Ching-Nam Hang, Siu-Wai Ho, and Chee-Wei Tan. 2023. Natural language generation and understanding of big code for AI-assisted programming: A review. *Entropy* 25, 6 (2023), 888.
[44] Eric Yu. 2024. Agent-Oriented Modeling for the Age of AI: Nine Pivots Toward a Reconceptualization of Requirements Engineering. In *Social Modeling Using the i* Framework: Essays in Honour of Eric Yu.* Springer, 207–227.
[45] Olaf Zawacki-Richter, Victoria I Marín, Melissa Bond, and Franziska Gouverneur. 2019. Systematic review of research on artificial intelligence applications in higher education–where are the educators? *International journal of educational technology in higher education* 16, 1 (2019), 1–27.