Published online on the page: https://jurnal.ilcagam.org/index.php/jsti

J S T I
Jurnal Serumpun Teknik Informatika
| ISSN (*Online*) 3123-965X | DOI: https://doi.org/10.66485/jsti.v1i3.34

Software Engineer Competency Framework in the Era of Generative AI: A Literature Review

Muhamad Anggun Novembra[1,*]

[1]*Islamic University of West Sumatera, Padang,* Indonesia

Article Information

Article History:
Submit: 18 Juli 2026
Revision: 28 Juli 2026
Accepted: 17 Agustus 2026
Published: 30 Agustus 2026

Keywords

Generative Artificial Intelligence; Software Engineer Competencies; Prompt Engineering; Competency Framework;

Correspondence

E-mail: mhdanggoen@uisb.ac.id

ABSTRACT

Generative artificial intelligence (GenAI) technologies such as Claude Code, ChatGPT, and GitHub Copilot are fundamentally reshaping software development practices, shifting the core activities of software engineers from direct code authoring toward validation, orchestration, and architectural reasoning. This paradigm shift raises fundamental questions: what competencies do software engineers require to collaborate effectively alongside GenAI, and how do these requirements vary across career stages? A focused literature review informed by the Systematic Literature Review principles of Kitchenham & Charters (2007) and the mapping study guidelines of Petersen et al. (2015) was conducted to address these questions. Findings were synthesized into a three-pillar competency model: foundational technical competencies augmented by AI tool literacy and prompt engineering; cognitive-analytical skills characterized by intensified critical code review, systems thinking, and AI-generated logic verification; and meta-skills encompassing AI governance, ethical judgment, and continuous adaptability, all of which exhibit significant differentiation across junior, mid-level, and senior software engineers. Furthermore, this study identifies a critical research gap: competency evolution at the senior engineering tier remains substantially under-researched compared to junior and mid-level stages. These findings offer practical implications for software organizations and educational institutions in redesigning competency development pathways in the GenAI era

1. Introduction

Generative artificial intelligence (GenAI) technology, particularly large language models (LLMs) such as Claude Code, ChatGPT, Antigravity, GitHub Copilot, and others, has transformed software engineering (SE) practices at an unprecedented pace. Early research has extensively examined the technical impact of GenAI on code quality and developer productivity; however, its broader implications for the specific competencies required by software engineers remain fragmented and insufficiently synthesized.

Unlike previous waves of automation that primarily targeted routine and low-skilled labor, GenAI directly affects high-income cognitive occupations. Experimental studies demonstrate that GenAI yields substantial productivity gains in professional writing (Noy & Zhang, 2023), complex knowledge work (Dell'Acqua et al., 2023), and specifically software coding tasks (Peng et al., 2023). While democratizing software creation on one hand, it potentially erodes the market value of routine, entry-level coding skills on the other. The software industry is transitioning from "writing code" to "cognitive orchestration"—where the primary value proposition of a software engineer shifts toward architectural oversight, systems design, and rigorous validation of AI outputs. This aligns with empirical evidence showing that developers actively utilizing AI increasingly engage in diverse, higher-order tasks beyond routine coding (Kam et al., 2025).

This fundamental shift raises pivotal questions regarding the precise competencies software engineers must cultivate and how these needs diverge across career stages. To date, existing literature has not systematically synthesized these evolving competency requirements into a coherent, career-differentiated framework. This study aims to address this research gap through three core research questions: (1) What technical, cognitive, and meta-skills do software engineers need to work effectively alongside GenAI? (2) How do these competency demands differ across career stages (junior, mid-level, senior)? and (3) What are the practical implications of this competency framework for organizations, educational institutions, and individual software engineers navigating the transition to an AI-augmented era?

2. Literature Review

Prior research examining the impact of GenAI on the labor market has largely operated at the macroeconomic level without dissecting the granular competency shifts within the software engineering profession (Salari et al., 2025). Other investigations focus on technical and methodological adaptations within the SE lifecycle (code generation, testing, documentation) without systematically addressing how individual developer competencies must evolve (Mastropaolo et al., 2025), even though recent research roadmaps emphasize positioning humans as central agents within AI-era software engineering rather than passive end-users (Abrahão et al., 2025).

From an educational standpoint, studies reveal a pronounced disconnect between industry practices and higher education curricula: software organizations are rapidly adopting LLMs across their development pipelines, whereas computer science curricula remain comparatively rigid, with initial integration largely confined to introductory programming courses (Sengul et al., 2024). Kirova et al. (2024) corroborate this disparity, observing that while public discourse extensively discusses general AI integration in education, dedicated investigations into the implications of LLMs for software engineering education remain scarce. They advocate for a holistic educational paradigm integrating technical skills, ethical awareness, and adaptive learning strategies. This need is reinforced by Ozkaya et al. (2024) and MacNeil et al. (2023), who highlight the direct implications for computing educators and students. When routine coding tasks—which historically served as foundational scaffolding for novice skill acquisition—are automated by GenAI, the pedagogical efficacy of traditional learning mechanisms becomes compromised.

Furthermore, organizational orchestration frameworks (Maruping & Matook, 2020), which conceptualize software development as the coordinated orchestration of distributed agents and tools rather than isolated individual coding efforts, provide valuable theoretical grounding for this transformation. In the GenAI context, "orchestration" shifts from coordinating human team members to orchestrating human-AI collaboration, wherein software engineers steer and critically validate AI-generated artifacts.

Synthesizing these research streams highlights a clear void: no existing framework systematically integrates technical, cognitive, and meta-skills for software engineers in the GenAI era while delineating them across career trajectories. This study is designed to bridge this critical gap.

3. Research Method

This study employs a focused literature synthesis informed by the Systematic Literature Review (SLR) (Kitchenham & Charters 2007) and the systematic mapping study guidelines (Petersen et al. 2015).

The candidate literature corpus was compiled based on three search concept blocks: (1) GenAI/LLMs ("Generative AI", "GenAI", "Large Language Model", "AI assistant", "code generat*",

"ChatGPT", "GitHub Copilot"), (2) Software engineering domain ("Software Engineer", "Developer", "Software Development", "SE practic*"), and (3) Competency dimensions ("Skill", "Competenc*", "Prompt Engineering", "AI literacy"). The aggregated collection yielded 541 unique records following deduplication, serving as the empirical baseline for source selection.

The inclusion and prioritization criteria were defined as follows: (a) explicit focus on software engineers, developers, or programmers; (b) explicit examination of GenAI/LLMs; (c) substantive discussion of competencies, skills, or training requirements; (d) primary studies (secondary studies were excluded from core evidence to prevent double counting, though cited contextual literature was retained in the Introduction/Literature Review, consistent with Sengul et al., 2024); and (e) publication between 2022 and 2025, with exceptions for foundational labor theory and methodology literature.

From the 541 candidate papers, titles and abstracts were screened using large language model assistance (Claude, Anthropic) against the defined inclusion and exclusion criteria. The adoption of AI in this preliminary screening phase follows established precedents for LLM-assisted literature reviews (Scherbakov et al., 2025) while strictly adhering to human oversight and transparency standards outlined in AI evidence synthesis guidelines (Flemyng et al., 2025). This screening produced an initial classification: 141 potential core evidence studies, 99 contextual/background studies, 32 foundational papers, 202 records requiring manual verification (predominantly lacking abstracts), and 67 excluded studies. From the potential core evidence candidates, 17 studies directly addressing specific competency claims were fully verified via detailed abstract and text analysis and cited directly in the core findings, supplemented by 6 foundational methodological/theoretical works and supporting literature on meta-skills dimensions, resulting in a total of 29 references in the final bibliography (see Figure 1).

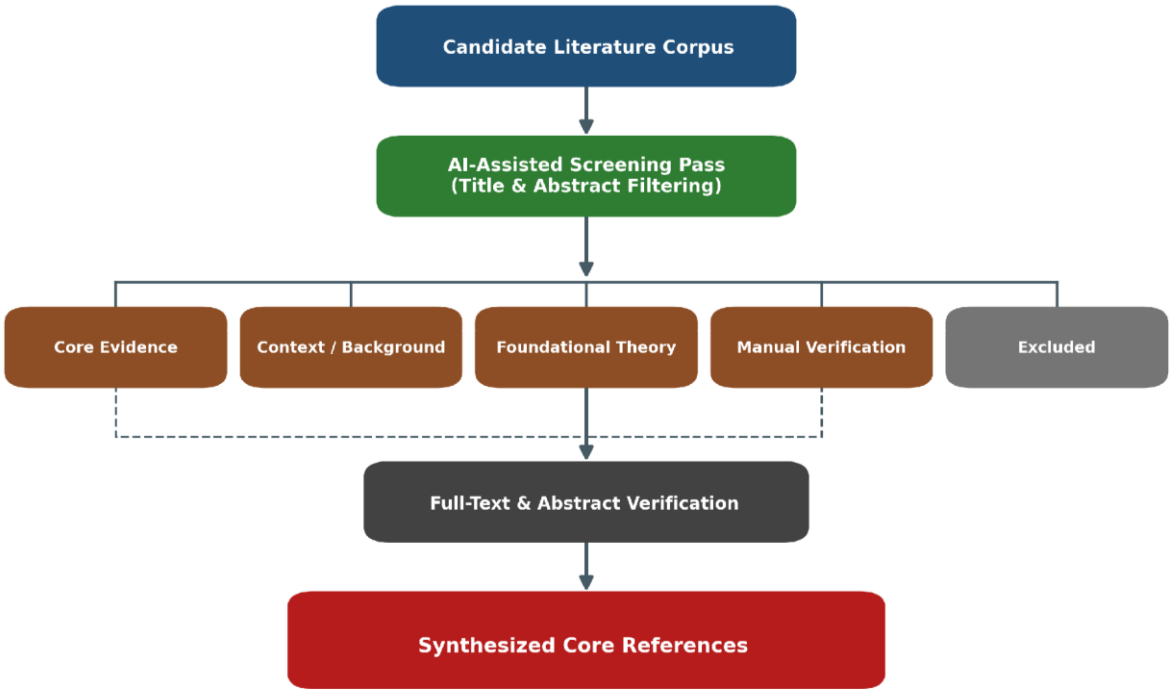

Figure 1. Literature Source Selection and Screening Workflow

4. Result & Discussion

To structure the Software Engineer Competency Framework in the GenAI Era (Figure 2) systematically, this study implemented a qualitative thematic synthesis methodology adapted from Thomas & Harden and empirical software engineering synthesis guidelines (Kitchenham & Charters, 2007; Petersen et al., 2015). The synthesis followed three systematic steps.

First -- Data Extraction and Initial Coding: Each verified primary study was analyzed to extract empirical findings regarding emerging developer tasks, newly demanded industry skills, LLM output failure modes (e.g., hallucinations, security vulnerabilities, bias), and cognitive risks such as cognitive offloading and superficial understanding among novice programmers.

Second -- Thematic Aggregation into Three Conceptual Pillars: Initial codes were inductively clustered into three holistic pillars: (a) Technical Competencies, covering direct operational interaction with AI tools and intelligent system programming; (b) Cognitive-Analytical Skills, representing higher-order critical thinking to audit, verify, and architect computational solutions; and (c) Meta-Skills, encompassing regulatory governance, professional ethics, and lifelong cognitive adaptability.

Third -- Career-Stage Mapping: Competencies within each pillar were mapped hierarchically across career maturity levels: Junior (0–3 years: foundational mastery and rigorous validation of AI outputs), Mid-Level (4–10 years: tool orchestration, system integration, and domain modeling), and Senior (10+ years: strategic system architecture leadership, ethical governance, and organizational mentoring).

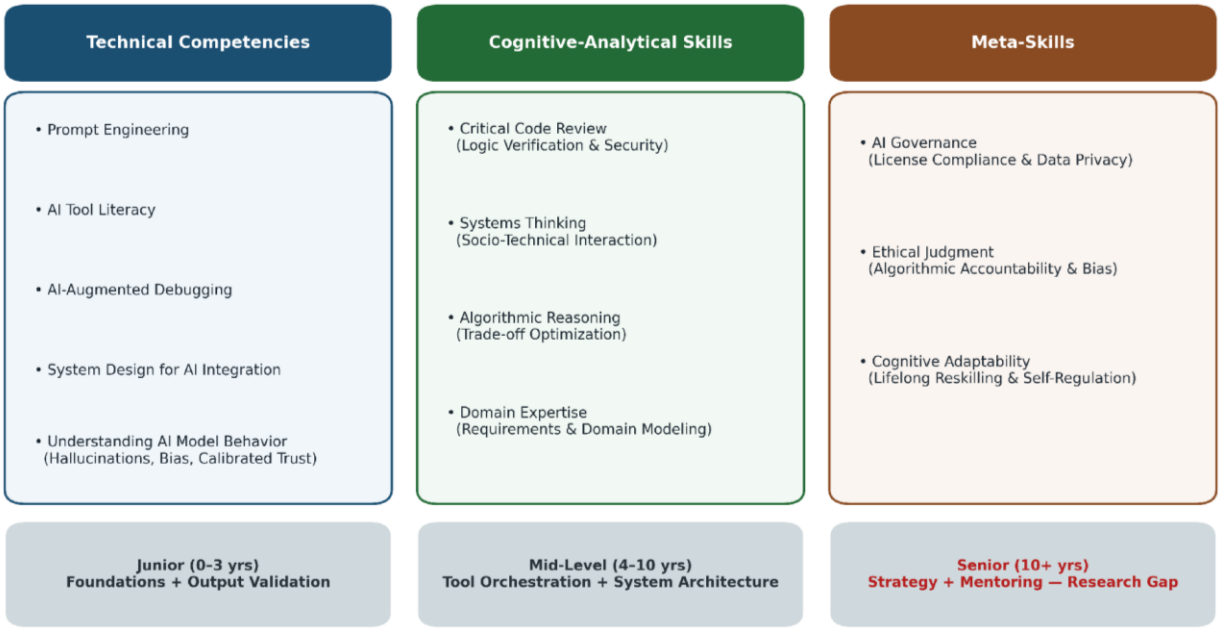

Figure 2. Software Engineer Competency Framework in the GenAI Era

4.1. Technical Competencies in the AI Era

The reviewed studies identify five emerging technical competency categories driven by GenAI adoption in software engineering practice (Table 1), spanning entry-level requirements to senior-level architectural responsibilities.

Table 1. Emerging Technical Competencies in the AI Era

Competency	Definition	Career Stage Requirement
Prompt Engineering	Ability to formulate effective, unambiguous prompts to generate correct and efficient code (Ehsani et al., 2025)	Junior and above
AI Tool Literacy	Understanding capabilities, constraints, and appropriate use cases for specific GenAI tools (López-Fernández & Vergaz, 2024)	Junior and above
AI-Augmented Debugging	Debugging workflows that blend AI-driven fault localization with human causal reasoning (Long et al., 2025; Khaliq et al., 2022)	Mid-Level and above
System Design for AI Integration	Architecting modular systems that leverage AI agents while maintaining performance, security, and reliability (Husen et al., 2024; Nahar et al., 2022)	Mid-Level and above

Understanding AI Model Behavior	Comprehending hallucinations/confabulations (Reinhard et al., 2025), algorithmic bias (De Martino et al., 2025), and establishing calibrated trust in AI output quality (Devalla & Yogix, 2023)	Senior

4.2. Transformation of Cognitive Skills

Beyond novel technical proficiencies, several established cognitive competencies in software engineering have undergone significant shifts in focus and intensification (Table 2)

Table 2. Cognitive Skill Transformations in the AI Era

Competency	Traditional Focus	Transformation in the AI Era
Critical Code Review	Stylistic compliance, syntax adherence, and local optimization	Rigorous verification of logical correctness, edge-case robustness, and security in AI-generated code (Zhu et al., 2025)
Systems Thinking	Component-level interaction and modular boundaries	Comprehension of human-AI socio-technical interactions as a foundation for career sustainability in ML/AI environments (Chuang et al., 2024)
Algorithmic Reasoning	Manual optimization techniques and algorithm design	Evaluating algorithmic trade-offs to critically verify, benchmark, and refine AI recommendations (Moradi Dakhel et al., 2023)
Domain Expertise	Deep specialized vertical knowledge	Explicit requirements elicitation and domain modeling as indispensable human-centric processes in AI system development (Ahmad et al., 2023)

4.3. Meta-Skills in the AI Era

In addition to technical and cognitive abilities, the increasing autonomy of GenAI necessitates a third foundational pillar: meta-skills. Meta-skills represent higher-order transversal capabilities that govern how engineers act responsibly, ethically, and adaptively amidst disruptive technological change (Table 3).

Table 3. Dimensions of Software Engineer Meta-Skills in the AI Era

Competency	Definition & Operational Focus	Career Stage Relevance	Key References
AI Governance	Ability to audit open-source licenses in generated code, track code provenance and cleanliness, mitigate software supply chain vulnerabilities, and safeguard corporate proprietary data privacy.	Mid-Level to Senior	Lu et al. (2022); Hou et al. (2024)
Ethical Judgment	Active deliberation on algorithmic accountability, auditing representation bias in training models, mitigating systemic discrimination, and enforcing Responsible-AI-by-Design principles.	All levels (Essential for Senior)	De Martino et al. (2025); Lu et al. (2022)
Cognitive Adaptability & Agility	Continuous reskilling amidst rapid LLM lifecycle shifts, metacognitive awareness to prevent cognitive offloading, and autonomous self-directed learning regulation.	All levels (Critical for Junior)	Chuang et al. (2024); Kirova et al. (2024)

These three meta-skill dimensions serve crucial operational roles in modern software engineering : first; AI Governance: GenAI adoption introduces substantial intellectual property and compliance challenges. Engineers must track code provenance to verify whether model-generated code incorporates copyleft-licensed snippets or infringes proprietary copyrights (Hou et al., 2024). Furthermore, preventing corporate data leakage to public training endpoints requires the rigorous application of verified Responsible AI architectural patterns (Lu et al., 2022); second, Ethical Judgment: As intelligent systems assume automated decision-making roles, software engineers serve as primary ethical gatekeepers (De Martino et al., 2025). Practitioners must actively identify and mitigate training data

biases that may cause disparate impact across user demographics. Embedding ethics from requirements engineering through testing constitutes a professional imperative to ensure socio-technical reliability (Lu et al., 2022). third; Cognitive Adaptability and Agility: The accelerated pace of AI advancement demands lifelong learning. Rather than relying solely on ephemeral programming syntax, engineers must cultivate conceptual flexibility to embrace emerging paradigms (Chuang et al., 2024). For junior developers, this meta-skill is vital in resisting cognitive offloading—excessive reliance on AI suggestions that undermines the development of fundamental computational intuition (Kirova et al., 2024).

4.4. Competency Differentiation Across Career Stages

Competency requirements vary significantly across career tiers. At the junior level (0–3 years), foundational SE knowledge (algorithms, debugging, system design) remains paramount, integrated alongside AI tool literacy with an emphasis on validating AI outputs. The primary challenge is acquiring deep conceptual mastery when tools handle surface tasks, mitigating the risk of an "illusion of competence" where novices lack the expertise to filter erroneous AI solutions (Moradi Dakhel et al., 2023).

At the mid-level (4–10 years), priorities shift toward integrating traditional expertise with orchestration skills and AI system integration. Developers transition into subject-matter experts guiding tool configuration, facilitating growth toward architecture or mentorship roles.

At the senior level (10+ years), strategic AI adoption, enterprise architecture decisions, and cross-functional mentoring take precedence. However, empirical studies investigating senior-level competency evolution remain sparse, marking this as the most critical research gap identified.

4.5. Practical Implications

For organizations, the risk of an "illusion of competence" among junior software engineers indicates that providing access to GenAI tools cannot be equated with the competence to validate their outputs. Onboarding pathways that traditionally relied on routine coding tasks as a primary learning vehicle risk losing their pedagogical utility when those tasks are automated. Organizations should implement structured mentorship programs that explicitly train developers in verification and orchestration rather than mere tool usage.

For educational institutions, the documented curricular lag (Sengul et al., 2024)—where GenAI literacy remains concentrated in introductory programming courses—signals an urgent need to expand integration into advanced courses that emphasize validation, system design, and domain reasoning. The pedagogical feasibility of this approach is supported by empirical precedents, such as the documented integration of ChatGPT as an interactive learning scaffold in database administration courses (López-Fernández & Vergaz, 2024). Assessment methodologies must also adapt from evaluating code authoring from scratch toward assessing the ability to audit, benchmark, and orchestrate AI-generated software artifacts.

For individual software engineers, the proposed framework serves as an actionable self-assessment roadmap: junior engineers can deliberately cultivate output validation skills rather than merely generating prompts; mid-level engineers can steer professional development toward orchestration and architecture; and senior engineers are positioned to actively shape industry best practices given the current dearth of mature empirical precedents.

5. Conclusions

Software engineering stands at a pivotal juncture: GenAI does not displace the profession but elevates it up the value chain from routine code production to cognitive orchestration. This review synthesizes these evolving competency demands into a three-pillar model (technical, cognitive, and meta-skills) differentiated across career stages, identifying the "illusion of competence" at junior levels and the significant research void surrounding senior engineer evolution as key insights. Organizations

must redesign onboarding to emphasize output validation; educational institutions must broaden AI literacy beyond introductory programming; and practitioners can leverage this framework for career development.

Several limitations and directions for future research should be acknowledged: First, this study represents a focused literature synthesis informed by SLR principles rather than an exhaustive full-scale review; screening depth was prioritized over breadth. Second, as a single-author review without an independent second reviewer to calculate inter-rater reliability (Cohen's Kappa), the study adhered to Kitchenham & Charters' (2007) test-retest sample evaluation protocol to maintain consistency. Third, the synthesized corpus of 29 references provides an initial thematic baseline. Future investigations expanding full manual verification across larger corpora with independent multi-reviewer validation will further enrich and extend the proposed framework.

References

Abrahão, S., Grundy, J., Pezzè, M., Storey, M.-A., & Tamburri, D. A. (2025). Software engineering by and for humans in an AI era. ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3715111

Ahmad, K., Abdelrazek, M., Arora, C., Grundy, J., & Bano, M. (2023). Requirements elicitation and modelling of artificial intelligence systems: An empirical study. In Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023) (pp. 126–137). SCITEPRESS. https://doi.org/10.5220/0011842300003464

Chuang, S., Shahhosseini, M., Javaid, M., & Wang, G. G. (2024). Machine learning and AI technology-induced skill gaps and opportunities for continuous development of middle-skilled employees. Journal of Work-Applied Management. https://doi.org/10.1108/JWAM-08-2024-0111

De Martino, V., Voria, G., Troiano, C., Catolino, G., & Palomba, F. (2025). Examining the impact of bias mitigation algorithms on the sustainability of ML-enabled systems: A benchmark study. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2025.112458

Dell'Acqua, F., McFowland, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K. C., Rajendran, S., & Lakhani, K. R. (2023). Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4573321

Devalla, S., & Yogix, M. K. (2023). Building trust in AI: A simplified guide to ensure software quality. Journal of Soft Computing Paradigm, 5(3). https://doi.org/10.36548/jscp.2023.3.001

Ehsani, R., Pathak, S., & Chatterjee, P. (2025). Towards detecting prompt knowledge gaps for improved LLM-guided issue resolution. 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR). https://doi.org/10.1109/MSR66628.2025.00107

Flemyng, E., Noel-Storr, A., Macura, B., Gartlehner, G., Thomas, J., Meerpohl, J. J., Jordan, Z., Minx, J., Eisele-Metzger, A., Hamel, C., Jemioło, P., Porritt, K., & Grainger, M. (2025). Position statement on artificial intelligence (AI) use in evidence synthesis across Cochrane, the Campbell Collaboration, JBI, and the Collaboration for Environmental Evidence 2025. JBI Evidence Synthesis, 23(11), 2162–2166.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. ACM Transactions on Software Engineering and Methodology, 33(5), Article 130, 48 pages. https://doi.org/10.1145/3695988

Husen, J. H., Washizaki, H., Runpakprakun, J., Yoshioka, N., Tun, H. T., Fukazawa, Y., & Takeuchi, H. (2024). Integrated multi-view modeling for reliable machine learning-intensive software engineering. Software Quality Journal. https://doi.org/10.1007/s11219-024-09687-z

Kam, M., Miller, C., Wang, M., Tidwell, A., Lee, I. A., & Malyn-Smith, J. (2025). What do professional software developers need to know to succeed in an age of artificial intelligence? Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3696630.3727251

Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). Artificial intelligence in software testing: Impact, problems, challenges and prospect. arXiv. https://doi.org/10.48550/arXiv.2201.05371

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering (Version 2.3). EBSE Technical Report EBSE-2007-01, Keele University and University of Durham.

Kirova, V. D., Ku, C. S., Laracy, J. R., & Marlowe, T. J. (2024). Software engineering education must adapt and evolve for an LLM environment. Proceedings of the 55th ACM Technical Symposium on Computer Science Education. https://doi.org/10.1145/3626252.3630927

Long, X., Tan, X., Zhu, Y., Jiang, J., & Zhang, L. (2025). Understanding and enhancing CS students' interaction experience with AI coding assistant tools. ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3785479

López-Fernández, D., & Vergaz, R. (2024). Adoption and impact of ChatGPT in computer science education: A case study on a database administration course. AI, 5(4). https://doi.org/10.3390/ai5040114

Lu, Q., Zhu, L., Xu, X., Whittle, J., Douglas, D., & Sanderson, C. (2022). Software engineering for responsible AI: An empirical study and operationalised patterns. In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '22) (pp. 241–242). ACM. https://doi.org/10.1145/3510457.3513063

Maruping, L. M., & Matook, S. (2020). The evolution of software development orchestration: Current state and an agenda for future research. European Journal of Information Systems, 29(5), 443–457. https://doi.org/10.1080/0960085X.2020.1831834

Mastropaolo, A., Escobar-Velásquez, C., & Linares-Vásquez, M. (2025). From triumph to uncertainty: The journey of software engineering in the AI era. ACM Transactions on Software Engineering and Methodology, 34(5), Article 131, 34 pages. https://doi.org/10.1145/3709360

Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., & Jiang, Z. M. (2023). GitHub Copilot AI pair programmer: Asset or liability? Journal of Systems and Software, 203. https://doi.org/10.1016/j.jss.2023.111734

MacNeil, S., Kim, J., Leinonen, J., & Denny, P. (2023). The implications of large language models for CS teachers and students. Proceedings of the 54th ACM Technical Symposium on Computer Science Education. https://doi.org/10.1145/3545947.3573358

Nahar, N., Zhou, S., Lewis, G., & Kästner, C. (2022). Collaboration challenges in building ML-enabled systems: Communication, documentation, engineering, and process. In Proceedings of the 44th International Conference on Software Engineering (ICSE '22) (pp. 413–425). ACM. https://doi.org/10.1145/3510003.3510209

Noy, S., & Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. Science, 381(6654), 187–192. https://doi.org/10.1126/science.adh2586

Ozkaya, I., Schmidt, D., & Hilton, M. (2024). Generative AI and software engineering education. Software Engineering Institute, Carnegie Mellon University. https://doi.org/10.58012/AHC6-GK69

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot. arXiv. https://doi.org/10.48550/arXiv.2302.06590

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Reinhard, P., Li, M. M., Fina, M., & Leimeister, J. M. (2025). Fact or fiction? Exploring explanations to identify factual confabulations in RAG-based LLM systems. Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3706599.3720249

Salari, N., Beiromvand, M., Hosseinian-Far, A., Habibi, J., Babajani, F., & Mohammadi, M. (2025). Impacts of generative artificial intelligence on the future of labor market: A systematic review. Computers in Human Behavior Reports, 18, 100652. https://doi.org/10.1016/j.chbr.2025.100652

Scherbakov, D., Hubig, N., Jansari, V., Bakumenko, A., & Lenert, L. A. (2025). The emergence of large language models as tools in literature reviews: A large language model-assisted systematic review. Journal of the American Medical Informatics Association, 32(6), 1071–1086. https://doi.org/10.1093/jamia/ocaf063

Sengul, C., Neykova, R., & Destefanis, G. (2024). Software engineering education in the era of conversational AI: Current trends and future directions. Frontiers in Artificial Intelligence, 7, 1436350. https://doi.org/10.3389/frai.2024.1436350

Zhu, Q., Wang, Z., Zhuang, Y., Chen, G., Luo, H., Wu, M., Chen, H., & Liu, Z. (2025). An automated code review framework based on BERT and Qianwen large model. https://doi.org/10.1109/ccai65422.2025.11189422