IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Evaluating GenAI in the Software Development Lifecycle: Theory vs. Developer Practice

M.Santhoshkumar[1], V.Divya[2]

[1]Research Scholar, School of Computing Science, Vels Institute of Science, Technology & Advanced Studies, Pallavaram, Chennai, India.

[2]Assistant Professor, Department of Computer Applications (UG), School of Computing Science, Vels Institute of Science, Technology & Advanced Studies, Pallavaram, Chennai, India.

Abstract: Generative Artificial Intelligence (GenAI) rapidly transforms software engineering, yet existing research remains fragmented across individual tasks in the Software Development Lifecycle. This study integrates a systematic literature review with a survey of 65 software developers. The results show that GenAI exerts its highest impact in design, implementation, testing, and documentation, where over 70 % of developers report at least halving the time for boilerplate and documentation tasks. 79 % of survey respondents use GenAI daily, preferring browser-based Large Language Models over alternatives integrated directly in their development environment. Governance is maturing, with two-thirds of organizations maintaining formal or informal guidelines. In contrast, early SDLC phases such as planning and requirements analysis show markedly lower reported benefits. In a nutshell, GenAI shifts value creation from routine coding toward specification quality, architectural reasoning, and oversight, while risks such as uncritical adoption, skill erosion, and technical debt require robust governance and human-in-the-loop mechanisms.

Keywords Generative Artificial Intelligence · Software Engineering · Large Language Models · AI-assisted Coding · IT Governance · Agile Development

1 Introduction

The rapid adoption of Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs), is substantially transforming the Software Development Life Cycle (SDLC). Chui et al. [2023] identify software engineering as one of the domains with the highest economic impact potential from GenAI. AI-assisted tools such as GitHub Copilot are increasingly embedded in professional software engineering environments, supporting activities that span the entire lifecycle—from requirements analysis to architectural design, code generation, testing, debugging, documentation, and maintenance.

GenAI refers to a class of machine learning systems capable of producing novel content (such as text, images, code, or sound) by learning patterns from large datasets and generating contextually coherent outputs [Goodfellow et al., 2016]. Prominent examples include LLMs such as ChatGPT, which process and generate natural language. While these systems do not replace human judgment or accountability, they might augment developer capabilities, potentially enabling faster iteration cycles, improved efficiency, and higher software quality. At the same time, its adoption raises questions related to maintainability, security, intellectual property, transparency, regulatory compliance, and long-term developer skill development.

Despite growing interest, the literature remains fragmented: Studies typically focus on aspects related to isolated SDLC phases, while comprehensive assessments across the full lifecycle with empirical data remain scarce. Research agendas such as Feuerriegel et al. [2024] call for more empirical investigations into how GenAI transforms organizational work practices and, specifically, how such systems affect software development productivity. To address these gaps, we derive the following research questions.

RQ1: How are GenAI tools currently used by software developers?

RQ2: How is the impact of GenAI perceived across the phases of the SDLC?

RQ3: What governance mechanisms are implemented to manage the adoption of GenAI in software development departments?

RQ4: What are the perceived risks associated with GenAI in software development?

2 Methodology for Literature Analysis

This section outlines the methodology used to identify and select relevant studies for the literature analysis presented in Section 3. Our literature selection process follows established methodological guidelines for conducting systematic literature reviews in Information Systems research [Webster and Watson, 2002, Okoli, 2015]. Following the typology proposed by Paré et al. [2015], who distinguish nine literature review types, ours can be characterized as a scoping review: Its primary goal is to identify and structure the current body of research on GenAI across the SDLC, rather than to aggregate quantitative data or to develop new theoretical propositions.

The selection criteria included peer-reviewed journal articles, conference proceedings, preprints, and technical reports published by established consulting and research organizations. Extending the evidence base beyond academic publications is in line with Okoli [2015], who acknowledges non-peer-reviewed sources as part of the searchable evidence base in systematic reviews.

The primary databases searched were Scopus, IEEE Xplore, the ACM Digital Library, SpringerLink, arXiv, and Google Scholar. These sources were chosen to ensure broad coverage of software engineering, information systems, and applied AI research. Search queries combined general and domain-specific keywords. Core search terms included “AI-assisted coding”, “GitHub Copilot”, “Generative AI”, “large language models”, “agile framework” and “software development tools”. These were complemented by phase-specific terms corresponding to the SDLC, such as “requirements engineering”, “software design”, “code generation”, “testing”, “debugging”, “maintenance” and “project management”. In addition, selected industry reports from consulting and technology firms such as McKinsey, Boston Consulting Group (BCG), Deloitte, and Accenture were included, as these publications often provide early insights into technological adoption patterns and practical challenges that precede academic research.

3 Literature Review

The rapid advancement of GenAI is increasingly transforming software development practices across the SDLC. Among AI-assisted development tools, conversational LLMs such as ChatGPT and assistants integrated directly in the development environment like GitHub Copilot, Amazon CodeWhisperer, JetBrains AI Assistant, and Tabnine have become widely adopted, supporting tasks such as code generation, explanation, and review [Sergeyuk et al., 2025].

Empirical studies and field research consistently report productivity gains across development activities, particularly in code optimization (approximately 60 %), bug fixing (26 %), and documentation support (12 %) [Collante et al., 2025]. Evidence further suggests that less experienced developers benefit disproportionately from AI assistance, achieving higher relative productivity improvements [Brynjolfsson et al., 2023, Dell’Acqua et al., 2023, Ng et al., 2024]. At the same time, studies indicate that code quality generally remains stable or improves despite reduced development time [Shihab et al., 2025, Yadav and Mondal, 2025]. However, correctness cannot be guaranteed and long-term risks remain, making human validation and oversight essential [Yetiştiren et al., 2023, Atif et al., 2025]. Beyond individual productivity, industry reports highlight broader organizational implications, including accelerated prototyping, shorter time-to-market, and shifting accountability toward product and project leadership [Deniz et al., 2023,

Gnanasambandam et al., 2025]. The following subsections synthesize the literature across the phases of the SDLC, before addressing agile software development as well as the risks and roadblocks associated with GenAI adoption.

3.1 GenAI in the Software Development Lifecycle

3.1.1 Planning.

In the planning phase, GenAI supports the creation of foundational project artifacts such as scope definitions, timelines, milestone structures, role descriptions, communication plans, and risk assessments [Barcaui and Monat, 2023, Hughes et al., 2025, Maggoo et al., 2025]. AI systems are also used to conduct market analyses, synthesize competitive information, and formulate product goals and visions [Lechner et al., 2025, Dell'Acqua et al., 2023]. Cost estimation and story point approximation have likewise been explored as application areas [Wagner and Wagner, 2024].

Empirical evidence suggests considerable productivity gains during project initiation and planning, including faster setup of new projects and improved alignment between technical teams and management [Aramali et al., 2025, Brynjolfsson et al., 2023, Hughes et al., 2025]. GenAI is typically deployed via structured prompts and increasingly integrated into project management tools such as Jira, Confluence, or Microsoft Project [Barcaui and Monat, 2023, Ng et al., 2024]. These capabilities contribute to early identification of risks and structured project preparation.

3.1.2 Requirements Analysis.

During requirements analysis, GenAI assists in drafting, refining, and validating functional and technical requirements. LLMs are used to generate and elaborate user stories and use cases based on high-level project documentation [Cico et al., 2023, Pinto et al., 2023, Wei et al., 2025] and to create preliminary architectural drafts and structured feature specifications [Glever et al., 2024, Gong et al., 2025, Nitin, 2024]. AI-assisted approaches reduce conceptualization time and accelerate the creation of prototypes, including visually oriented UI/UX artifacts [Wei, 2024, Wei et al., 2025, Gnanasambandam et al., 2025]. Tools such as Figma AI and related plugins seem promising implementations for rapid prototyping [Petridis et al., 2024]. Furthermore, AI systems can analyze requirement documents to identify inconsistencies, ambiguities, and missing elements, enabling earlier error detection and reducing downstream rework [Gong et al., 2025, Porter et al., 2025]. Improved consistency between business and IT stakeholders has also been reported [Pinto et al., 2023, Brynjolfsson et al., 2023].

3.1.3 Design and Implementation.

The design and implementation phase represents the most extensively studied application domain for GenAI. AI-assisted coding tools such as GitHub Copilot provide real-time code completion, refactoring suggestions, boilerplate generation, and context-aware optimization [Peng et al., 2023, Yetiştiren et al., 2023, Solohubov et al., 2023, Glever et al., 2024, Gerdemann et al., 2024]. They also support automatic documentation generation, including code comments and API documentation [Atif et al., 2025, Pinto et al., 2023, Ndiaye et al., 2025], as well as recommendations to improve code structure and maintainability [Gong et al., 2025, Zhong et al., 2025].

Experimental and field studies consistently report increased development speed [Peng et al., 2023, Cui et al., 2024, Shihab et al., 2025, Struever et al., 2025, Deniz et al., 2023]. Routine tasks are particularly affected, with automation reducing manual effort for standardized coding activities [Solohubov et al., 2023]. Several analyses also indicate improvements in review efficiency and defect reduction, although human oversight remains essential [Collante et al., 2025, Yetiştiren et al., 2023, Ndiaye et al., 2025]. Implementation typically occurs via integration into Integrated Development Environments (IDEs) such as Visual Studio Code or IntelliJ, often combined with secure APIs or isolated models to address data protection and compliance requirements [Ng et al., 2024, Ebert and Louridas, 2023].

3.1.4 Testing and Integration.

In the testing and integration phase, GenAI is used to automatically generate unit, integration, and end-to-end test cases [Atif et al., 2025, Glever et al., 2024, Maggoo et al., 2025, Zhong et al., 2025]. Additional applications include the creation of mock objects and synthetic test data [Yetiştiren et al., 2023, Gong et al., 2025], suggestions for missing test paths and coverage gaps [Cico et al., 2023, Nitin, 2024], and AI-assisted security analysis [Ding et al., 2024, Pearce et al., 2025, Ndiaye et al., 2025]. The literature documents reductions in manual testing effort for routine scenarios and broader, faster test coverage [Kathiresan, 2024, Solohubov et al., 2023, Zhong et al., 2025] as well as earlier detection of edge cases and integration issues [Yetiştiren et al., 2023, Kathiresan, 2024]. In practice, GenAI is often integrated into CI/CD pipelines and connected to test frameworks or embedded directly into development environments for automated code analysis and test case generation [Ng et al., 2024, Pinto et al., 2023].

3.1.5 Operation and Maintenance.

In the operation and maintenance phase, GenAI supports the analysis of logs, error messages, and stack traces, providing diagnostic insights and suggesting potential remediation steps based on learned patterns [Ng et al., 2024, Gong et al., 2025, Zhong et al., 2025]. It is also used to explain, summarize, and restructure poorly documented or complex legacy code, improving maintainability and knowledge transfer [Nitin, 2024, Atif et al., 2025, Maggoo et al., 2025].

Reported effects include faster issue detection and resolution, improved traceability, and enhanced resilience to staff turnover due to automated documentation and knowledge preservation [Ng et al., 2024, Pinto et al., 2023, Nitin, 2024]. Integration commonly occurs within incident and log management platforms such as Jira, ServiceNow, or Grafana, and in sensitive environments through locally deployed or access-controlled AI systems to mitigate compliance risks [Ng et al., 2024, Ebert and Louridas, 2023].

3.2 GenAI in Agile Software Development

GenAI influences the dynamics of agile development practices beyond productivity gains. The literature reports high perceived usefulness and satisfaction in agile teams when using AI-assisted tools [Geyer et al., 2025] and accelerated task completion, leading to shorter iteration cycles and more frequent releases [Zhang et al., 2024, Ulfsnes et al., 2024, Bahi et al., 2024]. By automating routine coding, documentation, and testing tasks, GenAI enables teams to focus on value-generating increments, reinforcing core agile principles such as rapid feedback and continuous delivery. At the same time, role expectations shift: Product owners, developers, and scrum masters increasingly engage in strategic, creative, and evaluative activities, while routine tasks are partially delegated to AI systems [Diebold, 2025, Bahi et al., 2024, Nasir and Hussain, 2024]. The literature recommends adapting role profiles and strengthening AI-related competencies, particularly critical evaluation skills in AI-supported environments.

3.3 Risks and Roadblocks in the Adoption of GenAI

Despite substantial productivity gains, the literature identifies a range of short- and long-term risks associated with the integration of GenAI into software development. In the short term, concerns relate to reliability, quality control, and security. AI-generated code suggestions are frequently erroneous, with manual correction efforts averaging approximately ten minutes per identified defect [Yetiştiren et al., 2023]. A significant proportion of outputs requires post-editing before productive use [Ziegler et al., 2022]. Developers often accept AI-generated snippets with limited scrutiny, suggesting superficial quality control practices [Ziegler et al., 2022, Chen et al., 2021]. Perceived productivity improvements do not always align with objective activity metrics, raising the risk of overestimating efficiency gains [Stray et al., 2025]. Security vulnerabilities represent a further concern: coding assistants can be susceptible to prompt injection attacks, potentially leading to data contamination, unintended code execution, or data leakage [Liu et al., 2025, Cotroneo et al., 2024, Norton et al., 2025].

Long-term risks are more structural. Continuous reliance on AI-generated suggestions may erode deep technical knowledge and increase dependency on AI systems [Ng et al., 2024, Brynjolfsson et al., 2023, Ding et al., 2025]. Automatically generated code can introduce opaque dependencies and architectural inconsistencies, contributing to technical debt [Ebert and Louridas, 2023, Atif et al., 2025, Anderson et al., 2025, Moreschini et al., 2026]. Even syntactically correct code may contain subtle semantic or security-relevant weaknesses [Yetiştiren et al., 2023, Atif et al., 2025]. Organizational dependence on specific vendors poses additional strategic risks, including technology lock-in and compliance exposure [Ng et al., 2024].

Overall, the literature indicates that GenAI exerts measurable influence across all phases of the SDLC, with the strongest effects observed in design, implementation, and testing. However, these gains are neither uniform nor automatically realized—short-term limitations in reliability and security, long-term risks such as skill erosion and technical debt as well as organizational roadblocks may constrain or offset productivity improvements. Effective governance frameworks, structured human-in-the-loop validation, and deliberate organizational adaptation remain critical to translating GenAI's technical capabilities into lasting benefits.

4 Survey Design

To complement the literature analysis with practitioner perspectives, we conduct a standardized online survey informed by the TAM [Davis, 1989, Venkatesh and Davis, 2000]. TAM provides a widely used framework for examining how users perceive the usefulness and adoption of new technologies in organizational contexts. The survey investigates how software practitioners use GenAI tools, how they assess productivity effects across different SDLC phases, how governance mechanisms for GenAI adoption are implemented in practice, and which risks practitioners associate with the use of GenAI.

Data were collected using a self-developed online questionnaire. The instrument comprised five sections: (1) demographic and professional background variables (age, education, role, team size); (2) primary development methodology (agile, traditional, hybrid) and degree of adherence; (3) GenAI tool usage behavior and frequency;
(4) assessments of productivity, quality, and security implications across SDLC phases; and (5) time-based efficiency estimations and open-ended questions on perceived opportunities and risks. A pretest ($n = 3$) was conducted to refine wording and improve the distinction between process model categories.

5 Survey Results

In the following we present the findings of our quantitative online survey (n = 65). Although the sample exhibits a comparatively young age structure, it demonstrates a strong professional profile. The majority of respondents are industry practitioners, including 30 professional software developers, one team lead, and four product owners. Notably, ten participants report more than ten years of professional experience, indicating that the sample includes senior expertise despite its youthful demographic composition.

Figure 5.1: Development methodology and age distribution

With regard to development methodologies, the agile approaches Scrum and Kanban constitute the largest share at about 42 %. Approximately 34 % report operating without a clearly defined process model, followed by 17 % of respondents who use a hybrid approach with both agile and traditional (waterfall-based) methodologies. The comparably smallest group with about 8 % of respondents work in formalized or plan-driven ("Traditional") environments.

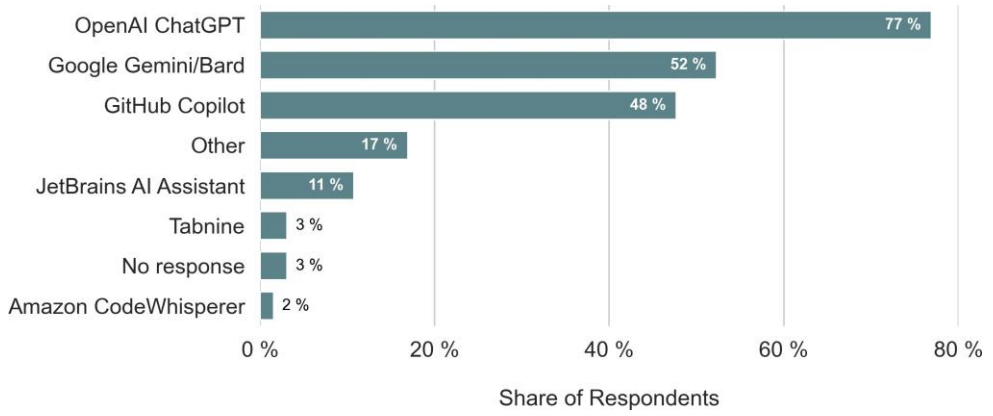

Figure 5.2: GenAI tool usage

As Figure 5.2 presents, the survey reveals widespread integration of GenAI into daily development activities. Approximately 79 % of respondents report using AI tools at least once per day, indicating that GenAI has become a routine component of professional practice. In terms of tools, browser-based LLMs dominate. ChatGPT is used by 77 % of participants, making it the most frequently adopted tool, followed by Google Gemini/Bard at 52 %. GitHub Copilot, which combines IDE-integrated code completion with a conversational chat interface, is used by about 48 % of respondents. IDE-integrated coding assistants such as JetBrains AI Assistant (11 %), Tabnine (3 %), and Amazon CodeWhisperer (2 %) remain marginal by comparison. 17 % of respondents reported using other tools while 3 % did not respond to the tool-related question. This distribution suggests that developers favor dialog-based interaction with general-purpose LLMs over specialized IDE-integrated alternatives.

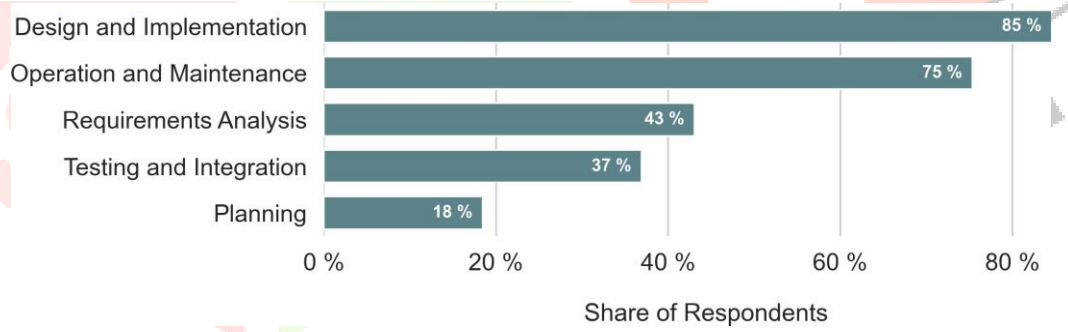

Figure 5.3: Impact of GenAI by SDLC phase

Figure 5.3 displays the distribution of responses regarding the SDLC phases in which participants perceived the greatest utility of GenAI (multiple choice). 85 % of respondents identify “Design and Implementation” as the phase with the strongest perceived benefit, followed by “Operation and Maintenance” at 75 %. Earlier lifecycle phases receive notably lower ratings, with “Requirements Analysis” at 43 %, “Testing and Integration” at roughly 37 %, and “Planning” at 18 %. Although the low value for “Planning” might be attributed to the high proportion of junior software developers in the sample, none of the Product Owners and only 5 % of respondents with more than ten years of experience report a perceived impact in this phase, suggesting the opposite.

To quantify these benefits at the task level, respondents were asked to estimate the time required for six common development activities when using GenAI in relation to working without AI support (see Figure 5.4). The results show that the strongest effects are reported for writing boilerplate code and documentation, where 72 % and 69 % of respondents, respectively, estimate at least halving the required time. These two tasks fall under design and implementation as well as operation and maintenance; the two SDLC phases also rated highest in Figure 5.3. Writing new code shows comparably more moderate positive effects, with the largest group (37 %) reporting a notable reduction to approximately 75 %. Creating unit tests also shows substantial gains among the respondents, although the high "cannot assess" rate (28 %) suggests that not all respondents regularly perform this activity. Understanding unfamiliar or legacy code produces the most polarized responses, with 29 % reporting extreme speedup to less than 25 % but also the highest "no change" rate (11 %) across all tasks. Debugging yields the weakest perceived savings and the highest share of respondents reporting increased effort (6 %), consistent with the cognitively complex nature of this task. Together, those findings confirm the primary role of GenAI as a coding assistant that accelerates repetitive and standardized development tasks, while also suggesting substantial perceived value in operation and maintenance activities such as debugging, log analysis, and legacy code comprehension.

The qualitative responses largely reinforce the quantitative findings and provide deeper insight into perceived opportunities, risks, and anticipated role changes associated with GenAI in software development. Across roles and experience levels, the dominant theme is efficiency. Participants repeatedly emphasize time savings through automation of repetitive and low-complexity tasks such as boilerplate code generation, documentation, test creation, debugging support, and code explanation. Many respondents describe GenAI as a "sparring partner" that accelerates learning, facilitates onboarding into new technologies, and supports rapid prototyping. Experienced developers highlight its value in reviewing novel code, interfacing with unfamiliar systems, and exploring alternative solution paths. Several comments state that AI reduces "toil" and allows developers to focus more on architecture, system design, and complex problem solving.

At the same time, concerns are substantial and remarkably consistent. The most frequently mentioned risks are uncritical adoption of AI-generated code, loss of deep technical understanding, and data protection and security vulnerabilities. Many respondents warn against "blind copying" and emphasize that AI outputs often appear syntactically correct but may contain logical, architectural, or security flaws. Senior practitioners stress that experience remains indispensable for evaluating AI-generated artifacts. A recurring theme is the danger of cognitive offloading: if AI is used as a shortcut rather than as a support tool, individual learning curves and long-term problem-solving capabilities may deteriorate. Data security—particularly the risk of exposing sensitive or company-specific information to external models—is also repeatedly cited as a major concern.

Regarding the future of the profession, most participants do not anticipate the disappearance of software development but rather a structural transformation of the role. A common expectation is a shift from pure code writing toward architectural thinking, requirement formulation, quality assurance, and orchestration of AI systems. Several respondents predict a polarization effect: junior roles may decline in number, while senior and architect-level competencies become more valuable. Others foresee new hybrid roles, such as AI supervisors. While a minority expresses skepticism or uncertainty about long-term effects, the prevailing view is that GenAI will become a standard tool, and that developers who fail to integrate it into their workflows may fall behind.

6 Conclusion

We examined the impact of GenAI on the SDLC by integrating a systematic literature review with empirical evidence from a cross-sectional survey of 65 software developers. The findings consistently demonstrate that AI-assisted coding tools exert substantial influence across multiple SDLC phases, with the most pronounced and empirically validated effects observed in design, implementation, testing, and documentation. We determine that prior research reports measurable productivity gains, reductions in routine workload, and generally stable or improved code quality when human validation mechanisms remain in place.

The survey results largely corroborate these findings. Respondents identify implementation, boiler plate code, and documentation as the domains with the highest perceived value, confirming the role of GenAI as a productivity multiplier in syntactically structured and repetitive tasks. In contrast, less structured and evaluative tasks such as planning, requirements analysis, debugging, and testing show considerably lower perceived productivity gains. Beyond phase-specific effects, the study highlights structural and organizational dimensions of AI integration. GenAI has become deeply embedded in daily workflows, yet security concerns remain moderately elevated, indicating that developers are aware of potential risks despite widespread adoption. The governance analysis reveals that a majority of organizations have already established either formal AI policies or at least informal usage guidelines, yet, a meaningful minority report regulatory ambiguity or limited awareness of existing policies, pointing to communication and implementation gaps.

Importantly, the findings suggest that GenAI does not eliminate the need for expertise but instead shifts value creation within the SDLC. As routine coding activities become increasingly automated, higher-order compe-tencies, such as requirements precision, architectural reasoning, critical validation, and governance oversight, gain importance. Experience level moderates perception: early-career professionals report particularly strong perceived learning benefits, whereas experienced practitioners tend to frame AI primarily as an efficiency-enhancing tool. This dynamic underscores the importance of structured training and human-in-the-loop mechanisms to prevent overreliance and ensure sustainable skill development.

References

[1] Michael Chui, Roger Roberts, Lareina Yee, Eric Hazan, Alex Singla, Kate Smaje, Alex Sukharevsky, and Rodney Zemmel. The economic potential of generative AI: The next productivity frontier. Technical report, McKinsey, June 2023. URL https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/ the-economic-potential-of-generative-ai-the-next-productivity-frontier.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, Cambridge, MA, USA, November 2016. ISBN 978-0-262-03561-3. URL https://mitpress.mit.edu/9780262035613/deep-learning/.

[3] Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. Generative AI. *Business & Information Systems Engineering*, 66:111–126, 2024. ISSN 2363-7005. doi:10.1007/s12599-023-00834-7.

[4] Fred Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *Management Information Systems Quarterly*, 13(3), December 1989. ISSN ISSN 0276-7783/ISSN 2162-9730. URL https://aisel.aisnet.org/misq/vol13/iss3/6.

[5] Viswanath Venkatesh and Fred D. Davis. A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science*, 46(2):186–204, February 2000. ISSN 0025-1909, 1526-5501. doi:10.1287/mnsc.46.2.186.11926. URL https://pubsonline.informs.org/doi/10.1287/mnsc.46.2.186.11926.

[6] Jane Webster and Richard T. Watson. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2):xiii–xxiii, 2002. URL http://www.jstor.org/stable/4132319.

[7] Chitu Okoli. A guide to conducting a standalone systematic literature review. *Communications of the Association for Information Systems*, 37, 2015.

[8] Guy Paré, Marie-Claude Trudel, Mirou Jaana, and Spyros Kitsiou. Synthesizing information systems knowledge: A typology of literature reviews. *Information & Management*, 52(2):183–199, March 2015. ISSN 03787206. doi:10.1016/j.im.2014.08.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0378720614001116.

[9] Yair Levy and Timothy J. Ellis. A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. *Informing Science: The International Journal of an Emerging Transdiscipline*, 9:181–212, 2006. ISSN 1547-9684, 1521-4672. doi:10.28945/479. URL https://www.informingscience.org/Publications/479.

[10] Kim Fröhnel, Gabriel Benedikt Thomas Adams, and Rüdiger Zarnekow. Facing the Multifaceted Impact of Fake Reviews: A Comprehensive Literature Review. *Wirtschaftsinformatik 2024 Proceedings*, January 2024. URL https://aisel.aisnet.org/wi2024/39.

[11] Agnia Sergeyuk, Yaroslav Golubev, Timofey Bryksin, and Iftekhar Ahmed. Using AI-based coding

assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology*, 178:107610, February 2025. ISSN 0950-5849. doi:10.1016/j.infsof.2024.107610. URL https://www.sciencedirect. com/science/article/pii/S0950584924002155.

[12] Antonio Collante, Samuel Abedu, SayedHassan Khatoonabadi, Ahmad Abdellatif, Ebube Alor, and Emad Shihab. The Impact of Large Language Models (LLMs) on Code Review Process, August 2025. URL http://arxiv.org/abs/2508.11034. arXiv:2508.11034 [cs].

[13] Erik Brynjolfsson, Danielle Li, and Lindsey R Raymond. Generative AI at work. 2023. URL https:
[14] //www.nber.org/system/files/working_papers/w31161/w31161.pdf.

[15] Fabrizio Dell'Acqua, Edward McFowland, Ethan R. Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R. Lakhani. Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality. *SSRN Electronic Journal*, 2023. ISSN 1556-5068. doi:10.2139/ssrn.4573321. URL https:
[16] //www.ssrn.com/abstract=4573321.

[17] Kevin KB Ng, Liyana Fauzi, Leon Leow, and Jaren Ng. Harnessing the Potential of Gen-AI Coding Assistants in Public Sector Software Development, September 2024. URL http://arxiv.org/abs/2409.17434. arXiv:2409.17434 [cs].