Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.068137

ARTICLE

Redefining the Programmer: Human-AI Collaboration, LLMs, and Security in Modern Software Engineering

Elyson De La Cruz*, **Hanh Le**, **Karthik Meduri**, **Geeta Sandeep Nadella*** **and Hari Gonaygunta**

School of Computer and Information Sciences, University of the Cumberlands, Williamsburg, KY 40769, USA

*Corresponding Authors: Elyson De La Cruz. Email: elyson.delacruz@ucumberlands.edu;
Geeta Sandeep Nadella. Email: gnadella3853@ucumberlands.edu

Received: 21 May 2025; Accepted: 06 August 2025; Published: 23 September 2025

ABSTRACT: The rapid integration of artificial intelligence (AI) into software development, driven by large language models (LLMs), is reshaping the role of programmers from traditional coders into strategic collaborators within Industry 4.0 ecosystems. This qualitative study employs a hermeneutic phenomenological approach to explore the lived experiences of Information Technology (IT) professionals as they navigate a dynamic technological landscape marked by intelligent automation, shifting professional identities, and emerging ethical concerns. Findings indicate that developers are actively adapting to AI-augmented environments by engaging in continuous upskilling, prompt engineering, interdisciplinary collaboration, and heightened ethical awareness. However, participants also voiced growing concerns about the reliability and security of AI-generated code, noting that these tools can introduce hidden vulnerabilities and reduce critical engagement due to automation bias. Many described instances of flawed logic, insecure patterns, or syntactically correct but contextually inappropriate suggestions, underscoring the need for rigorous human oversight. Additionally, the study reveals anxieties around job displacement and the gradual erosion of fundamental coding skills, particularly in environments where AI tools dominate routine development tasks. These findings highlight an urgent need for educational reforms, industry standards, and organizational policies that prioritize both technical robustness and the preservation of human expertise. As AI becomes increasingly embedded in software engineering workflows, this research offers timely insights into how developers and organizations can responsibly integrate intelligent systems to promote accountability, resilience, and innovation across the software development lifecycle.

KEYWORDS: Human-AI collaboration; large language models; AI security; developer identity; ethical AI in software development; AI-assisted programming

1 Introduction

The software development landscape is undergoing a seismic shift with artificial intelligence (AI) integration, particularly through tools powered by large language models (LLMs) such as GitHub Copilot and ChatGPT. These tools offer real-time code generation, bug detection, and task automation, thereby redefining the traditional responsibilities of developers [1]. This study posits that developers are no longer just coders—they are shifting into oversight roles, collaborating with AI systems to generate and refine code, while taking on responsibility for reviewing outputs and ensuring correctness and security. As developers collaborate more frequently with intelligent systems, their role is evolving from writing code manually to managing, auditing, and optimizing AI-generated outputs.

This transformation is especially significant in Industry 4.0, which emphasizes smart manufacturing, cyber-physical systems, and autonomous decision-making. AI-enhanced development tools are deployed within these environments to enable faster prototyping, adaptive maintenance, and real-time data processing [2]. Consequently, the modern programmer is no longer a solitary technician but an interdisciplinary collaborator contributing to intelligent, secure, and responsive industrial infrastructures.

While AI tools enhance efficiency, they also introduce challenges in ensuring the integrity and reliability of code. Developers must now assess potential vulnerabilities, data biases, and the ethical implications of AI-generated software, which requires technical acumen and critical thinking [3]. These concerns are particularly pressing in Industry 4.0 systems, where compromised code can lead to large-scale operational and safety failures.

Moreover, this AI-driven transition has psychological and educational implications. Many developers report concerns about skill redundancy and diminished autonomy, while others embrace AI to offload mundane tasks and focus on creative problem-solving [4]. These mixed sentiments underscore the need for support structures, including updated training curricula and transparent AI usage policies, to navigate the new normal in software development.

In response to this evolving reality, this study adopts a qualitative phenomenological approach to explore how developers perceive and adapt to their changing roles. Drawing from interviews with practicing IT professionals, we investigate how AI tools reshape workflows, developer identity, and skill expectations. The findings aim to guide educators, organizations, and policymakers in preparing a workforce capable of thriving at the intersection of AI, Industry 4.0, and secure software engineering.

2 Research Objectives and Questions

2.1 Research Objectives

The convergence of machine learning models, notably Large language models (LLMs), and software engineering practices is profoundly reshaping the developer's role. Developers are increasingly expected to move beyond code generation to engage in decision-making, ethical oversight, and dynamic collaboration with AI agents. These evolving expectations are particularly significant in Industry 4.0, where human-AI synergy is essential for building secure, adaptive, and intelligent systems [5].

This study explores how software professionals experience and adapt to the collaborative, security-focused, and knowledge-intensive demands of AI-powered development environments. The research examines the benefits and the technical, psychological, and organizational challenges accompanying the integration of AI tools in professional workflows. This research contributes to an evidence-based understanding of how human-AI collaboration can enhance software development resilience, creativity, and responsibility [6,7].

The primary objectives of this research are as follows:

1. Investigate how AI tools, such as ChatGPT and GitHub Copilot, are reshaping the roles and responsibilities of software developers within AI-enhanced development environments.
2. Analyze how developers address new security and ethical risks introduced by LLM-generated code in critical Industry 4.0 contexts.
3. Explore how developers envision future human-AI collaboration models and what skills and structures are needed to sustain productive partnerships.

2.2 Research Questions

In pursuit of these objectives, the study is anchored in the following research questions, grounded in the qualitative phenomenological framework:

RQ1: How do developers perceive the evolving nature of their roles in AI-augmented software development environments?

RQ2: What strategies do developers employ to address security, reliability, and ethical concerns using LLMs in software engineering?

RQ3: How do developers anticipate the future of human-AI collaboration in shaping software development workflows, team dynamics, and professional growth?

These questions provide a holistic understanding of how the developer's identity and workflows adapt to AI-infused ecosystems, with implications for education, policy, and technology governance. As human-AI collaboration becomes central to modern software development, it is essential to understand how these partnerships function in practice and what cultural, ethical, and institutional frameworks are required to support them. Insights from this study can inform curriculum development in computer science education, helping to prepare students for hybrid roles that blend coding expertise with prompt engineering, ethical reasoning, and AI literacy. Moreover, findings may assist policymakers and industry leaders in crafting guidelines and organizational structures that ensure AI's responsible integration, fostering innovation and accountability in Industry 4.0 settings.

3 Literature Review

The accelerating integration of AI into software development has prompted significant academic inquiry into its implications for programming practices, professional roles, and human-computer collaboration. As AI tools become increasingly embedded in development environments, researchers have examined the opportunities and disruptions they introduce. This literature review synthesizes current research on the transformation of programmer roles, the theoretical frameworks explaining these shifts, the psychological and ethical challenges involved, and the evolving educational demands in a rapidly changing technological landscape. The review is organized into thematic subheadings to present a structured analysis of key trends and gaps in understanding the future of programming in an AI-dominated context.

3.1 The Rise of Large Language Models in Software Development

Large language models (LLMs), such as the core components in ChatGPT and GitHub Copilot, have catalyzed a new era in software development by enabling natural language interaction, automated code suggestions, and context-aware debugging. These AI tools are no longer passive assistants but are being positioned as collaborative agents in development workflows. Developers increasingly rely on these systems to reduce repetitive workload and streamline problem-solving processes in enterprise and open-source contexts [8].

GitHub Copilot, for instance, has shown tangible productivity benefits in open-source environments, with studies reporting up to a 6.5% increase in project output [7]. However, this efficiency gain comes with a trade-off, specifically, higher integration times due to the need for oversight, coordination, and review of AI-suggested contributions. This new addition of the AI-generated tools highlights a critical need for skill development in prompt engineering and code review practices tailored to LLM-driven workflows.

3.2 Developer Identity and Role Transformation

The introduction of AI-powered development tools has significantly transformed how software developers define their professional roles. No longer limited to coding tasks, developers are now expected to perform as AI orchestrators—designing prompts, validating outputs, and applying ethical judgment in real time [9]. These new responsibilities call for hybrid competencies, including technical fluency, critical thinking, and collaborative communication.

Recent foresight studies predict that by 2030, AI will fully integrate into development environments, altering developers' daily routines and expanding their responsibilities across system architecture, security, and lifecycle optimization [10]. In this envisioned future, developers will manage machines and cultivate adaptive learning systems that require human-AI co-evolution. This paradigm shift necessitates comprehensive upskilling initiatives and organizational changes to support the redefinition of the programmer's identity.

3.3 Security Challenges in LLM-Generated Code

While LLMs offer efficiency and automation, their use introduces complex code security and reliability challenges. A recent systematic review found that AI-generated code often includes security vulnerabilities more frequently than human-written code, ranging from improper input validation to misuse of insecure libraries [11]. These flaws pose serious risks in critical systems, such as industrial automation or financial platforms, where failure can have large-scale consequences.

The integration of AI-powered coding assistants into software development workflows has introduced both efficiency gains and new risks. While these tools can generate functional code quickly, they often lack an understanding of secure coding practices. As a result, AI-generated code may include subtle vulnerabilities that are not immediately apparent to developers, especially those who lack deep security expertise. This issue resides in software security, which is critical in areas such as healthcare, finance, and defense, where even minor oversights can lead to data breaches or system compromise [12].

Recent research highlights the extent of these risks [13], evaluated code produced by GitHub Copilot and found that a substantial percentage—up to 40% depending on the prompt—contained security flaws. These vulnerabilities included SQL injection, hardcoded secrets, weak cryptographic functions, and improper error handling. One reason for this is that Copilot and similar models are trained on public code repositories, which may themselves contain insecure or outdated practices. Since the models replicate patterns seen in their training data without fully understanding the context, they often propagate these poor practices into new code, putting applications at risk.

Explored how students interact with GitHub Copilot in educational environments [14]. The study revealed that learners generally viewed the tool as a means to increase coding efficiency. However, concerns emerged regarding their ability to fully comprehend the code generated by the AI, as well as the potential for excessive dependence on the tool. The researchers identified two primary patterns of engagement: while some students proactively leveraged Copilot to guide their problem-solving, others encountered difficulties when faced with inaccurate or misleading suggestions. The study underscored the cognitive and metacognitive demands introduced by AI-assisted programming, emphasizing the pedagogical challenges of integrating such technologies into the learning process.

To address these issues, scholars recommend integrating human-in-the-loop validation, static analysis tools, and multi-tiered testing frameworks. By combining AI-generated outputs with structured review processes, developers can retain the speed benefits of LLMs while maintaining software integrity. Nonetheless, there remains an urgent need for more robust security standards and training programs tailored to AI-augmented development environments.

3.4 Collaboration Models and AI-Human Synergy

The dynamics of human-AI collaboration are shifting from tool-based assistance to full-fledged partnership models. The Human-AI framework offers a comprehensive view of this evolution by emphasizing five collaborative principles: bidirectional learning, explainability, co-validation, shared goals, and mutual augmentation [15]. These principles help frame AI as a co-developer that can evolve alongside the human team.

Trust is also central to effective collaboration. The VIRTSI model maps human trust states, such as over-trust and under-trust, and their impact on productivity in AI-integrated workflows [16]. Empirical evaluations using ChatGPT in development tasks have shown that maintaining calibrated trust leads to more effective interaction, smoother handoffs, and fewer errors. Thus, fostering explainability and user confidence in AI systems is essential for sustainable integration in software teams.

3.5 Implications for Industry 4.0 and Future Research Direction

The convergence of LLMs, AI-assisted development, and Industry 4.0 technologies presents a powerful opportunity to modernize industrial systems through intelligent, secure, and adaptive software infrastructures. AI-powered coding enables rapid prototyping, real-time data processing, and autonomous system upgrades—key demands of smart factories and cyber-physical systems [17]. However, the fast pace of integration raises concerns about ethical design, workforce readiness, and infrastructure resilience.

Future research must explore how developers can be equipped to operate effectively within these intelligent ecosystems. Such inquiry includes studying the socio-technical interplay between human and AI collaborators, developing governance frameworks for ethical LLM use, and formulating AI literacy programs in engineering education. As AI becomes a central agent in software creation, its governance, transparency, and alignment with human values will determine the quality and sustainability of future digital infrastructures.

4 Methodology

This study employs hermeneutic phenomenology as its core methodological approach. It aims to explore and interpret the lived experiences of software developers working with AI-driven tools such as GitHub Copilot and ChatGPT. Hermeneutic phenomenology, rooted in the philosophical traditions of Heidegger and Van Manen, focuses on describing experiences and interpreting their meaning in context [18,19]. This approach is especially suited to investigating how developers construct meaning around their evolving roles, identities, and ethical responsibilities in AI-augmented environments.

The rationale for choosing hermeneutic phenomenology is twofold. First, it allows the researcher to access rich, personal narratives about human-AI collaboration, an area of increasing interest in computing and social science. Second, it supports the development of a nuanced understanding of the psychological, professional, and social transformations developers undergo when integrating AI systems into their daily work [20]. In contrast to purely descriptive or statistical methods, this interpretive lens provides the depth required to unpack the complexity of experience in contexts shaped by rapid technological change.

Three theoretical frameworks guided the interpretation of data in this study: Deskilling of Jobs Theory (DJT) [21], Job Characteristics Theory (JCT) [22], and Technology Threat Avoidance Theory (TTAT) [23]. DJT offered a lens to examine concerns about skill atrophy and the potential loss of professional autonomy as developers increasingly rely on large language models to automate coding tasks. JCT helped contextualize how the integration of AI impacts core job attributes, such as task variety, autonomy, and significance, and how these changes influence developers' motivation and job satisfaction in intelligent, Industry

4.0-aligned environments. Meanwhile, TTAT provided insight into developers' cognitive and emotional responses to navigating perceived risks associated with AI security, ethical uncertainties, and trust in LLM-generated code.

Fig. 1 illustrates the interconnections between the theoretical frameworks underpinning this study and the emergent themes. The network visualization demonstrates how the Distributed Justice Theory (DJT), Job Characteristics Theory (JCT), and Technology Threat Avoidance Theory (TTAT) inform different aspects of programmer identity and practice in AI-integrated environments. The thickness of connecting lines represents the strength of relationships between frameworks and themes, with JCT showing powerful connections to Human-AI Collaboration Models and TTAT strongly informing Security Challenges.

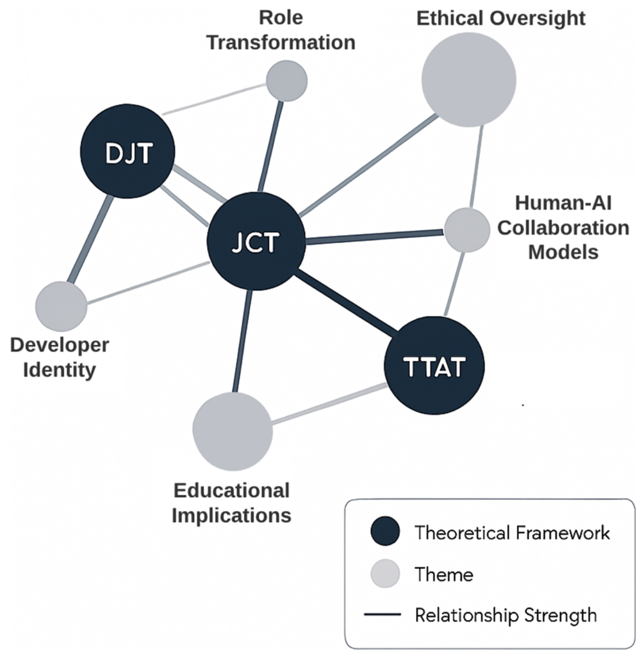


Figure 1: Theoretical framework network

Together, these frameworks offered complementary perspectives for understanding the evolving dynamics of human-AI collaboration in software engineering. By grounding qualitative insights in robust theoretical foundations, this approach enabled a multidimensional exploration of how developers are reconfiguring their skills, workflows, and professional identities in response to AI integration. It also facilitated analysis of the psychological tensions developers face, such as balancing efficiency with control, or innovation with security, thereby providing a holistic view of the challenges and opportunities presented by AI-augmented development in Industry 4.0 contexts.

4.1 Participant Selection and Data Collection

Participants were selected using purposive sampling, targeting professional developers with at least six months of experience using AI-assisted tools in real-world development projects. The sample included 15 software engineers from diverse organizational settings, including startups, enterprise environments, and remote freelance work. Semi-structured interviews were chosen as the primary data collection method, allowing participants to reflect deeply on their experiences while ensuring thematic consistency across sessions.

Each interview lasted approximately 45–60 min and was conducted in person or over a secure video conferencing platform. Questions were designed to probe participants' perceptions of their changing roles, security practices, team dynamics, and ethical considerations related to AI-generated code. All interviews were audio-recorded, transcribed verbatim, and anonymized for confidentiality.

4.2 Data Analysis

Data analysis followed a hermeneutic cycle described in contemporary qualitative research [24]. This process involved iterative movement between parts of the text (individual statements or themes) and the whole (overall meaning of the lived experience). Coding was conducted using thematic analysis, guided by both a priori themes from literature and emergent categories derived from participant narratives.

Particular attention was paid to "meaning structures" within the developers' language, such as metaphors of collaboration, tension between trust and oversight, and emotional reactions to automation. Themes were then synthesized to construct a cohesive narrative of the human experience in AI-augmented software development. To enhance validity, themes were discussed with a peer debriefer and member-checked with a subset of participants.

5 Findings

5.1 Developers Are Embracing New Hybrid Roles

Participants consistently described a shift in their professional identity from traditional coders to hybrid collaborators working alongside AI systems. GitHub Copilot and ChatGPT enhanced productivity by automating routine tasks, allowing developers to focus more on design thinking, problem-solving, and higher-level decision-making. This evolving relationship reflects broader industry trends in which developers act less as code authors and more as strategic orchestrators of AI workflows [25].

Several developers noted that their day-to-day work now requires skills beyond programming, including prompt engineering, evaluating AI suggestions, and ethical assessment. These findings support theoretical models predicting that developers will guide intelligent, Industry 4.0-aligned systems, emphasizing interdisciplinary fluency and AI literacy [10].

5.2 Security and Reliability Remain Core Developer Concerns

Despite the benefits of AI-generated code, participants expressed persistent concerns about code quality, security vulnerabilities, and the trustworthiness of LLM outputs. Many described experiences where Copilot or ChatGPT produced syntactically correct but logically flawed or insecure code. These instances of insecure code resulted in additional time spent reviewing, testing, and sometimes rewriting code to meet project or industry standards [26,27].

Several participants described the "false confidence trap", a tendency to over-trust AI-generated outputs due to their fluency and speed, only to discover subtle security issues later. These findings affirm the need for robust developer training in AI validation methods and highlight a significant gap in current development practices regarding secure deployment of AI-assisted code [28].

5.3 Emotional and Psychological Responses Are Mixed

Participants described a spectrum of emotional reactions to AI integration, ranging from enthusiasm to anxiety. Some welcomed the support, citing decreased burnout, more time for creative tasks, and a sense of empowerment through augmentation. Others reported fear of deskilling, loss of autonomy, or confusion about where their expertise was still needed [29].

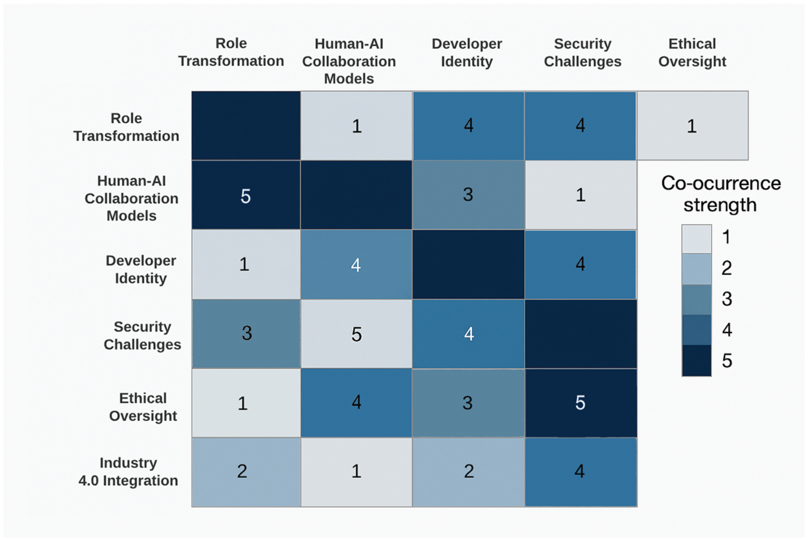
Notably, identity played a significant role in shaping emotional response. Developers who strongly identified as problem-solvers or mentors were more likely to view AI as a helpful tool. In contrast, those who valued deep technical immersion expressed greater ambivalence or resistance to AI tools, especially when outputs felt like "black boxes" [30].

5.4 Human-AI Collaboration Shapes New Learning Behavior

Adopting LLMs has led developers to adapt workflows and their learning and problem-solving behaviors. Participants reported using tools like ChatGPT as just-in-time knowledge sources, often bypassing traditional documentation in favor of conversational problem solving. This adoption has influenced how developers debug, experiment, and explore unfamiliar technologies [31].

This trend reflects a broader shift toward on-demand, AI-mediated learning in professional contexts. While some developers praised the accessibility of AI tools, others raised concerns about potential erosion in deep learning or long-term retention. These insights suggest that future educational and onboarding programs must account for the dual role of AI as both collaborator and knowledge mediator.

Fig. 2 presents a co-occurrence matrix visualizing the strength of relationships between the key themes identified in this study. The intensity of color and numeric values (1–5) indicate the degree of thematic interconnection, with darker cells representing stronger relationships. Notably, the strongest relationships (5) exist between Role Transformation and Human-AI Collaboration Models, Security Challenges and Ethical Oversight, and within each theme's self-relationship. This visualization highlights how security concerns are deeply intertwined with ethical considerations, while role transformation is fundamentally connected to emerging collaboration models.



	Role Transformation	Human-AI Collaboration Models	Developer Identity	Security Challenges	Ethical Oversight
Role Transformation		1	4	4	1
Human-AI Collaboration Models	5		3	1	
Developer Identity	1	4		4	
Security Challenges	3	5	4		
Ethical Oversight	1	4	3	5	
Industry 4.0 Integration	2	1	2	4	

Figure 2: Thematic Co-occurrence matrix

6 Discussion and Implications

6.1 Redefining the Developer Role in AI-Augmented Environments

The findings of this study reveal a paradigmatic shift in software development roles, where developers increasingly operate as curators of AI behavior rather than sole authors of code. This transformation aligns with broader models of hybrid intelligence, where human-AI collaboration emphasizes complementary strengths such as contextual reasoning by humans and pattern recognition by machines [32]. Developers must engage in critical evaluation, ethical filtering, and strategic decision-making as AI tools become embedded in everyday programming workflows.

This evolution calls for the redefinition of core competencies within software engineering. Future developers will require fluency in programming languages and prompt engineering, AI oversight, and trust calibration. These insights reinforce the need to redesign educational pathways and professional training to align with Industry 4.0 demands [33].

6.2 Security, Accountability, and the Ethics of AI Code Generation

A consistent concern emerging from the data is the reliability and security of AI-generated code. Developers are increasingly aware of the risks associated with over-reliance on LLM outputs, especially when these systems generate code with subtle bugs or security flaws. These challenges raise ethical questions around accountability—who is responsible when AI-generated code causes harm?—and the need for governance mechanisms that ensure human oversight and traceability [34].

To mitigate these risks, organizations must implement validation workflows that balance speed with safety, integrating static analysis tools and peer reviews into AI-assisted development pipelines. Moreover, institutions must define guidelines for the responsible use of AI in production systems, emphasizing transparency, explainability, and ethical risk assessments.

6.3 Implications for Workforce Development and Organizational Strategy

This study highlights an urgent need for workforce transformation strategies to prepare developers for AI-centric roles. Human resources departments, educational institutions, and tech leaders must co-create upskilling programs, AI-literacy training, and interdisciplinary collaboration models that enable smooth transitions into AI-augmented work environments [35]. Rather than viewing AI as a job eliminator, forward-looking strategies can position it as a co-worker that elevates human capabilities.

Additionally, this shift requires leadership that is not only technologically fluent but also attuned to emotional and cultural dynamics in AI adoption. The psychological impacts of AI, such as anxiety, trust gaps, or loss of identity, must be acknowledged and addressed through inclusive policy-making and participatory change management approaches.

Fig. 3 illustrates the temporal evolution of key concepts in programming practice from pre-AI integration to projected future states. This streamgraph visualization demonstrates how traditional programming concepts like Manual Coding and Individual Work have diminished in prominence, while emerging concepts such as AI Collaboration, Team Coordination, and System Architecture have gained significance. The visualization projects continued evolution toward AI Orchestration and Human-AI Teams, suggesting a fundamental transformation in how programming work is conceptualized and practiced.

6.4 Toward Human-AI Co-Creation in Industry 4.0 Systems

Finally, the findings emphasize the broader significance of human-AI co-creation in Industry 4.0 ecosystems. As smart manufacturing, cyber-physical systems, and predictive maintenance tools integrate AI coding assistants, software developers are emerging as the keystones of intelligent infrastructure. Their ability to mediate between AI capabilities and industrial needs will define the success of next-generation automation systems [32].

Designing these systems will require both technical and human-centered thinking, with developers shaping not only the logic of the machines but also the ethical and social architecture of their integration. As such, future programming must be understood not as isolated code production, but as a deeply collaborative, socio-technical act. Table 1 shows the impact of human-AI collaboration on developer roles and practices.

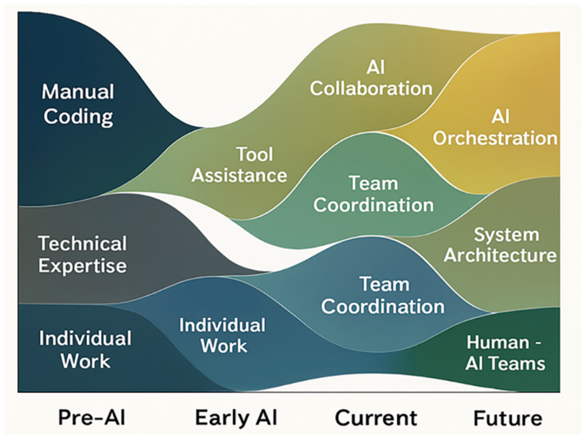


Figure 3: Conceptual development timeline

Table 1: Impact of human-AI collaboration on developer roles and practices

Domain	Traditional developer role	AI-augmented role (Post-LLMs)	Key implications
Code Generation	Manual coding, algorithm design	Prompt engineering, co-coding with AI	Emphasis shifts from syntax to strategy and supervision
Problem Solving	Step-by-step debugging, exploratory search	Rapid prototyping, pattern-based suggestions	Increased speed; reliance on LLM output validation
Security Practices	Static/manual code review	Continuous auditing of AI-generated code	New vulnerabilities and trust validation challenges
Knowledge Acquisition	Documentation, forums, trial and error	Real-time querying of LLMs	Accelerated learning, but risk of over-reliance on incomplete info
Identity & Autonomy	Self-identification as a coder, technical authority	Mixed emotions: pride, skepticism, fear of deskilling	Redefines autonomy; emotional adaptation needed
Collaboration & Workflow	Pair programming, CI/CD participation	Human-AI teaming, prompt iteration, output vetting	Collaboration expands to include non-human agents
Career Development	Tool mastery, language specialization	Interdisciplinary fluency (e.g., ethics, HCI, AI comprehension)	Training must evolve beyond code to include socio-technical literacy

7 Conclusions

This study highlights a fundamental transformation in software development, wherein developers are shifting from isolated code authors to strategic collaborators in AI-augmented ecosystems. Adopting tools such as GitHub Copilot and ChatGPT, powered by large language models, is reshaping how code is written and the roles, responsibilities, and identities of programmers in Industry 4.0 environments [36]. This shift presents opportunities for enhanced efficiency, real-time problem-solving, and continuous learning while demanding greater ethical oversight, security validation, and role clarity from software professionals.

Security remains a critical concern as developers balance the benefits of AI assistance with the risks of automation errors and code vulnerabilities. Our findings affirm that AI-generated outputs can boost productivity; they often require rigorous review and contextual interpretation to ensure software safety and integrity [14]. Trust calibration, transparent workflows, and continuous feedback loops between developers and AI tools will sustain secure and responsible AI use in critical development environments.

This transformation carries significant implications for education, workforce development, and organizational strategy. Institutions must prepare future developers with hybrid skill sets, including AI literacy, ethical reasoning, and prompt engineering. Meanwhile, organizations should establish AI governance structures and inclusive upskilling programs to foster resilience and adaptability in human-AI collaboration [6]. Industry 4.0 systems, in particular, demand developers who can act as ethical architects of intelligent infrastructures.

Ultimately, this research contributes to the emerging understanding of how AI can serve as a force multiplier rather than replace human intelligence when integrated ethically and strategically. Reframing programming as a co-creative process between humans and machines, we lay the groundwork for sustainable and resilient software practices. The future of programming will be shaped by what AI can do and how developers choose to collaborate with it.

8 Limitations and Future Scope

8.1 Limitations

While this study provides valuable insights into how developers adapt to AI-assisted programming tools, several limitations must be acknowledged. First, the sample size was limited to a small group of professional developers, which may not fully represent the diversity of experiences across geographies, industries, or organizational roles. As such, findings should be interpreted as context-specific and may not generalize to all development environments or demographic groups.

Second, the research was conducted within a narrow time frame during which generative AI tools rapidly evolve. This temporal limitation means that some developer perspectives may shift significantly as tools improve capability and integration. Additionally, self-reporting during interviews may have introduced subjective bias or social desirability effects, particularly concerning attitudes toward automation and job security.

8.2 Future Research

Future research should explore the longitudinal impact of AI-assisted programming on skill development, job satisfaction, and identity over time. The evaluation includes assessing whether early enthusiasm for tools such as Copilot translates into sustained productivity or leads to dependency and deskilling, particularly among junior developers [10]. Longitudinal, mixed-method studies could offer deeper insights into this evolution and its organizational consequences.

More research is needed on integrating AI tools within agile and DevOps workflows, including how teams coordinate trust, accountability, and security validation. Studies should also investigate the psychological and cognitive dynamics of human-AI teaming, examining how trust is built, when it breaks down, and how explainability and feedback loops can improve collaborative outcomes [37].

Finally, educational implications demand more attention. Future work should explore how universities and training programs can incorporate AI co-programming literacy, focusing on technical tools and ethics, critical thinking, and system design. Creating future-ready software professionals will require an interdisciplinary approach, bridging computing, human factors, and socio-technical systems thinking [38].

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Elyson De La Cruz: conceptualization, methodology, software, writing—original draft. Hanh Le and Geeta Sandeep Nadella: supervision, writing—review & editing. Karthik Meduri and Hari Gonaygunta: data curation, visualization. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated or analyzed during the current study are available from the corresponding author at a reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Hamza M, Siemon D, Akbar MA, Rahman T. Human-AI collaboration in software engineering: lessons learned from a hands-on workshop. In: 7th ACM/IEEE International Workshop on Software-intensive Business. Lisbon, Portugal: ACM; 2024. p. 7–14. doi:10.1145/3643690.3648236.
2. Huang Z, Fey M, Liu C, Beysel E, Xu X, Brecher C. Hybrid learning-based digital twin for manufacturing process: modeling framework and implementation. Robot Comput Integr Manuf. 2023;82(4):102545. doi:10.1016/j.rcim.2023.102545.
3. Staron M, Abraháo S, Serebrenik A, Penzenstadler B, Horkoff J, Honnenahalli C. Laws, ethics, and fairness in software engineering. IEEE Softw. 2025;42(1):110–3. doi:10.1109/MS.2024.3469488.
4. Shukla P, Bui P, Levy SS, Kowalski M, Baigelenov A, Parsons P. De-skilling, cognitive offloading, and misplaced responsibilities: potential ironies of AI-assisted design. In: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. Yokohama, Japan: ACM; 2025. p. 1–7. doi:10.1145/3706599.3719931.
5. Bécue A, Praça I, Gama J. Artificial intelligence, cyber-threats and industry 4.0: challenges and opportunities. Artif Intell Rev. 2021;54(5):3849–86. doi:10.1007/s10462-020-09942-2.
6. Sun X, Song Y. Unlocking the Synergy: increasing productivity through Human-AI collaboration in the industry 5.0 Era. Comput Ind Eng. 2025;200(3):110657. doi:10.1016/j.cie.2024.110657.
7. Song F, Agarwal A, Wen W. The impact of generative AI on collaborative open-source software development: evidence from GitHub copilot [Internet]. [cited 2025 Aug 1]. Available from: https://doi.org/10.2139/ssrn.4856935.
8. Barenkamp M, Rebstadt J, Thomas O. Applications of AI in classical software engineering. AI Perspect. 2020;2(1):1. doi:10.1186/s42467-020-00005-4.
9. Olla P, Elliott L, Abumeeiz M, Mihelich K, Olson J. Promptology: enhancing human-AI interaction in large language models. Information. 2024;15(10):634. doi:10.3390/info15100634.
10. Ciniselli M, Puccinelli N, Qiu K, Grazia L. From today's code to tomorrow's symphony: the AI transformation of developer's routine by 2030. arXiv:2405.12731. 2024. doi:10.48550/arXiv.2405.12731.
11. Ramírez LC, Limón X, Sánchez-García ÁJ, Pérez-Arriaga JC. State of the art of the security of code generated by LLMs: a systematic literature review. In: Proceedings of the 2024 12th International Conference in Software Engineering Research and Innovation (CONISOFT); 2024 Oct 28–Nov 1; Puerto Escondido, Mexico. Piscataway, NJ, USA: IEEE. p. 331–9. doi:10.1109/CONISOFT63288.2024.00050.
12. Wang R, Cheng R, Ford D, Zimmermann T. Investigating and designing for trust in AI-powered code generation tools. In: The 2024 ACM Conference on Fairness, Accountability, and Transparency. Rio de Janeiro, Brazil: ACM; 2024. p. 1475–93. doi:10.1145/3630106.3658984.
13. Pearce H, Ahmad B, Tan B, Dolan-Gavitt B, Karri R. Asleep at the keyboard? Assessing the security of GitHub copilot's code contributions. Commun ACM. 2025;68(2):96–105. doi:10.1145/3610721.

14. Prather J, Reeves BN, Denny P, Becker BA, Leinonen J, Luxton-Reilly A, et al. It's weird that it knows what I want: usability and interactions with copilot for novice programmers. ACM Trans Comput-Hum Interact. 2024;31(1):1–31. doi:10.1145/3617367.
15. Puerta-Beldarrain M, Gómez-Carmona O, Sánchez-Corcuera R, Casado-Mansilla D, López-de-Ipiña D, Chen L. A multifaceted vision of the human-AI collaboration: a comprehensive review. IEEE Access. 2025;13(4):29375–405. doi:10.1109/access.2025.3536095.
16. Virvou M, Tsihrintzis GA, Tsichrintzi EA. VIRTSI: a novel trust dynamics model enhancing artificial intelligence collaboration with human users-insights from a ChatGPT evaluation study. Inf Sci. 2024;675(3):120759. doi:10.1016/j.ins.2024.120759.
17. Grigorescu S, Zaha M. CyberCortex.AI: an AI-based operating system for autonomous robotics and complex automation. J Field Robot. 2025;42(2):474–92. doi:10.1002/rob.22426.
18. Ho KHM, Chiang VCL, Leung D. Hermeneutic phenomenological analysis: the 'possibility' beyond 'actuality' in thematic analysis. J Adv Nurs. 2017;73(7):1757–66. doi:10.1111/jan.13255.
19. Introna LD. Maintaining the reversibility of foldings: making the ethics (politics) of information technology visible. Ethics Inf Technol. 2007;9(1):11–25. doi:10.1007/s10676-006-9133-z.
20. Valový M, Buchalcevova A. The psychological effects of AI-assisted programming on students and professionals. In: Proceedings of the 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME); 2023 Oct 1–6; Bogotá, Colombia. Piscataway, NJ, USA: IEEE. p. 385–90. doi:10.1109/ICSME58846.2023.00050.
21. Ossadnik J, Muehlfeld K, Goerke L. Man or machine-or something in between? Social responses to voice assistants at work and their effects on job satisfaction. Comput Hum Behav. 2023;149(7):107919. doi:10.1016/j.chb.2023.107919.
22. Verma S, Singh V. Impact of artificial intelligence-enabled job characteristics and perceived substitution crisis on innovative work behavior of employees from high-tech firms. Comput Hum Behav. 2022;131(8):107215. doi:10.1016/j.chb.2022.107215.
23. Liang H, Xue Y. Understanding security behaviors in personal computer usage: a threat avoidance perspective. J Assoc Inf Syst. 2020;11(7):394–413. doi:10.17705/1jais.00232.
24. Watson S. Hermeneutic phenomenology. In: Khine MS, editor. Conceptual analyses of curriculum inquiry methodologies. Vol. 7. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2022. p. 616–27. doi:10.1002/9781119975144.ch7.
25. Maruping LM, Matook S. The evolution of software development orchestration: current state and an agenda for future research. Eur J Inf Syst. 2020;29(5):443–57. doi:10.1080/0960085x.2020.1831834.
26. Dunne M, Schram K, Fischmeister S. Weaknesses in LLM-generated code for embedded systems networki. In: 2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS); 2024 Jul 1–5; Cambridge, UK. Piscataway, NJ, USA: IEEE. p. 250–61. doi:10.1109/QRS62785.2024.00033.
27. Fawareh H, Al-Shdaifat HM, MAR, Fawareh FA, Khouj M. Investigates the impact of AI-generated code tools on software readability code quality factor. In: 2024 25th International Arab Conference on Information Technology (ACIT); 2024 Dec 10–12; Zarqa, Jordan. Piscataway, NJ, USA: IEEE. p. 1–5. doi:10.1109/ACIT62805.2024.10876897.
28. Pangavhane S, Raktate G, Pariane P, Shelar K, Wakchaure R, Kale JN. AI-augmented software development: boosting efficiency and quality. In: 2024 International Conference on Decision Aid Sciences and Applications (DASA); 2024 Dec 11–12; Manama, Bahrain. Piscataway, NJ, USA: IEEE. p. 1–5. doi:10.1109/DASA63652.2024.10836523.
29. Chen Y, Stavropoulou C, Narasinkan R, Baker A, Scarbrough H. Professionals' responses to the introduction of AI innovations in radiology and their implications for future adoption: a qualitative study. BMC Health Serv Res. 2021;21(1):813. doi:10.1186/s12913-021-06861-y.
30. von Eschenbach WJ. Transparency and the black box problem: why we do not trust AI. Philos Technol. 2021;34(4):1607–22. doi:10.1007/s13347-021-00477-0.
31. Alenezi M, Akour M. AI-driven innovations in software engineering: a review of current practices and future directions. Appl Sci. 2025;15(3):1344. doi:10.3390/app15031344.
32. Dellermann D, Ebel P, Söllner M, Leimeister JM. Hybrid intelligence. Bus Inf Syst Eng. 2019;61(5):637–43. doi:10.1007/s12599-019-00595-2.

33. Aranda-Jiménez JR, De-Pablos-Heredero C, Campos-García I, San-Martín J, Cosculluela-Martínez C. Digitalization and industry 4.0: what skills improve professional behaviours? J Ind Eng Manag. 2024;17(1):235. doi:10.3926/jiem.7091.
34. Shneiderman B. Bridging the gap between ethics and practice. ACM Trans Interact Intell Syst. 2020;10(4):1–31. doi:10.1145/3419764.
35. Samuels AB, Pelser AM. Transitioning from industry 4.0 to 5.0: sustainable supply chain management and talent management insights. SA J Hum Resour Manag. 2025;23:2874. doi:10.4102/sajhrm.v23i0.2874.
36. France SL. Navigating software development in the ChatGPT and GitHub copilot era. Bus Horiz. 2024;67(5):649–61. doi:10.1016/j.bushor.2024.05.009.
37. Yuan L, Gao X, Zheng Z, Edmonds M, Wu YN, Rossano F, et al. In situ bidirectional human-robot value alignment. Sci Robot. 2022;7(68):eabm4183. doi:10.1126/scirobotics.abm4183.
38. Abrahão S, Grundy J, Pezzè M, Storey MA, Tamburri DA. Software engineering by and for humans in an AI era. ACM Trans Softw Eng Methodol. 2025;34(5):1–46. doi:10.1145/3715111.