Responsibility and Ethical Practices in AI-Assisted Software Development

Alvin Wijaya*, Timo Poranen and Zheying Zhang

Faculty of Information Technology and Communication Sciences, Tampere University, Kalevantie 4, 33100 Tampere, Finland

Abstract

Artificial intelligence (AI) coding assistants such as GitHub Copilot, OpenAI Codex, and ChatGPT are increasingly embedded in professional software development workflows, offering real-time code suggestions and automation support. While empirical studies report measurable productivity gains, their integration raises significant ethical, legal, and security concerns. This paper examines how developers perceive responsibility, trust, and governance in the use of AI-assisted coding tools. The study employs a qualitative exploratory design, combining a short survey with seven semi-structured interviews involving developers across diverse professional roles. The findings reveal a distributed but developer-centered model of accountability. Participants consistently identified themselves as the final gatekeepers responsible for reviewing and deploying AI-generated code, while also emphasizing the roles of organizational governance and vendor safeguards in mitigating systemic risk.

A central theme is a "trust paradox": developers continue to rely on AI assistants despite recognizing recurring errors, security vulnerabilities, and intellectual property uncertainty. Trust was described as conditional and context-dependent, with greater reliance for boilerplate tasks and heightened scrutiny in security-sensitive contexts. Ethical awareness was strongest regarding security and intellectual property risks, while concerns related to bias were less prominent. The study identifies a gap between high-level ethical frameworks and everyday development practice. In the absence of formalized organizational guidance, developers rely on individual judgment and existing quality assurance mechanisms. The findings suggest that effective governance requires coordinated accountability structures combining human oversight, organizational policy, and enhanced transparency and security features at the vendor level. By grounding debates about AI ethics in practitioner experience, this research contributes empirical insight into how responsibility is interpreted and operationalized in AI-assisted software engineering.

Keywords

AI coding assistants, ethics of software development, trust paradox, responsibility, intellectual property, governance

CEUR-WS.org/Vol-4309/paper03.pdf

1. Introduction

AI coding assistants such as GitHub Copilot, OpenAI Codex, and ChatGPT are increasingly embedded in professional software development workflows. These systems generate real-time code suggestions, automate repetitive tasks, and support debugging and documentation. Empirical evidence shows measurable productivity gains [1, 2], and industry surveys indicate widespread adoption [3]. However, their integration raises significant ethical and technical concerns. Studies report frequent security vulnerabilities in AI-generated code [4, 5], while legal scholarship highlights unresolved intellectual property and licensing risks [6]. Research also documents embedded bias and uneven trust calibration, contributing to a "trust paradox" in which developers rely on AI outputs despite recognizing their limitations [7, 8]. These tensions foreground a central governance problem: responsibility. When AI-generated code causes harm whether through security flaws, bias, or copyright violations who is accountable? Regulatory and normative frameworks, including the UNESCO Recommendation on the Ethics of Artificial Intelligence [9] and the EU Artificial Intelligence Act [10], emphasize accountability and human oversight. Yet it remains unclear how responsibility is interpreted in everyday development practice. This study investigates developers' perceptions of ethical responsibility in AI-assisted coding.

The primary research question is:

TKTP 2026: Annual Doctoral Symposium of Computer Science, 8.—9.6.2026 Turku, Finland

*Corresponding author.

alvinwijaya159@gmail.com (A. Wijaya); timo.poranen@tuni.fi (T. Poranen); zheying.zhang@tuni.fi (Z. Zhang)

0000-0002-4638-0243 (T. Poranen); 0000-0002-6205-4210 (Z. Zhang)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-02

RQ1 – Responsibility: Who do developers think is responsible and accountable when ethical or technical problems arise from AI-generated code?

Supporting questions examine:

RQ2 – Ethical Practices: What ethical considerations (e.g., bias, security, copyright, accountability) influence how developers evaluate and use AI-generated code?

RQ3 – Future Outlook: What features, guidelines, or support systems could help ensure more ethical use of AI coding assistants in the future?

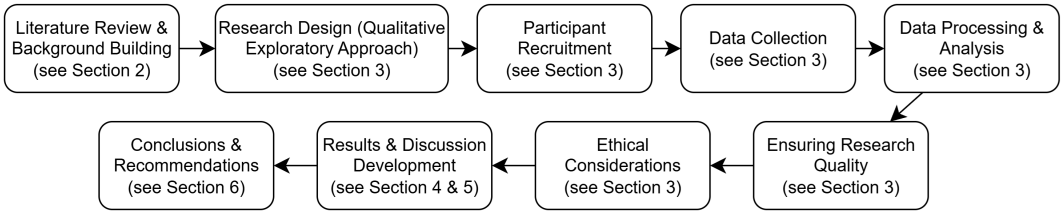


Figure 1: Overview of the research process.

Figure 1 illustrates the step-by-step methodological process. The research began with a literature review to establish theoretical grounding and identify key themes such as ethics, trust, and governance in AI-assisted coding (see Section 2). It then progressed through the design of a qualitative exploratory study [11, 12, 13], participant recruitment, and two-stage data collection involving a Google Form survey and semi-structured interviews (see Section 3). The collected data were processed through thematic analysis, ensuring rigor via credibility, dependability, confirmability, and transferability checks (see Section 3). Ethical considerations such as informed consent, confidentiality, and reflexivity were integrated throughout the research process (see Section 3). Finally, findings were synthesized into the results and discussion sections, leading to conclusions and recommendations for ethical AI practices in software development (see Sections 4–6). This paper builds on prior graduate research related to AI-assisted software development. [14].

2. Background

The rapid integration of generative artificial intelligence into software engineering has transformed how code is produced, reviewed, and maintained. AI coding assistants such as GitHub Copilot and ChatGPT are now embedded directly into development environments, offering real-time code completions, full-function generation, documentation support, and debugging suggestions. Controlled experiments demonstrate measurable productivity gains, with developers completing tasks significantly faster when assisted by AI [1]. Large-scale industry studies further report widespread adoption and high perceived usefulness, particularly for repetitive or boilerplate tasks [2, 3].

However, these productivity benefits are accompanied by persistent ethical and technical concerns. Security research shows that AI-generated code frequently contains vulnerabilities, including insecure cryptographic practices and injection risks [4, 5]. Developers often report spending additional time reviewing or correcting AI suggestions, suggesting that efficiency gains may be offset by quality assurance efforts. This dynamic reflects what scholars describe as a “trust paradox,” where practitioners rely on AI tools despite recognizing their limitations in correctness and security [7, 8].

Beyond technical risk, AI-generated code raises complex legal and ethical questions. Because large language models are trained on vast corpora of publicly available code, concerns persist regarding intellectual property (IP), licensing compatibility, and potential copyright infringement [6]. Developers may unknowingly incorporate code fragments that resemble licensed material, creating ambiguity about ownership and compliance. At the same time, empirical studies reveal that code generation models can reproduce social biases embedded in training data, potentially shaping downstream system behavior in subtle but consequential ways [15].

These concerns shift attention from performance metrics to responsibility and governance. Professional and regulatory frameworks including the ACM Code of Ethics, IEEE 7000-2021, UNESCO's AI Recommendation, and the EU AI Act emphasize accountability, transparency, and human oversight in AI systems [16, 17, 9, 10].

When AI-generated code results in a security breach, biased system output, or copyright dispute, the distribution of responsibility becomes ambiguous. Developers remain the final decision-makers who review and deploy code, but organizations select and integrate AI tools, and vendors design and train the underlying models. Existing literature has largely focused on performance, security, or legal implications, leaving a gap in understanding how practitioners themselves perceive ethical responsibility and how these perceptions influence their evaluation and use of AI-generated code. Understanding practitioner perspectives is important because ethical responsibility in software engineering is enacted through everyday development decisions.

This study therefore builds on prior research by examining how developers interpret responsibility when ethical or technical problems arise from AI-generated code, what ethical considerations shape their usage decisions, and what future mechanisms they believe would strengthen accountable AI-assisted development. By situating empirical findings within existing ethical and regulatory frameworks, the research contributes to a more grounded understanding of responsibility in the age of generative AI software engineering.

3. Methodology

This study employs a qualitative exploratory research design to investigate how developers perceive responsibility, ethical considerations, and future governance needs related to AI-generated code. Qualitative methods are appropriate when examining emerging phenomena that require understanding participants' interpretations, professional judgments, and contextual reasoning rather than measuring predefined variables [12, 11]. Because the primary research question focuses on how responsibility is attributed when ethical or technical harms arise from AI-generated code, an in-depth, interpretive approach was necessary. The design is structured around three research questions addressing responsibility (RQ1), ethical practices influencing code evaluation and usage (RQ2), and desired future safeguards or governance mechanisms (RQ3).

A purposive sampling strategy was used to recruit participants with direct experience using AI coding assistants. More specifically, the study employed criterion-based purposive sampling, in which participants are selected because they meet predefined criteria relevant to the research questions [18]. The primary inclusion criterion was direct professional experience using AI-assisted coding tools in a commercial or organizational software development context, with a minimum of six months of regular usage. Developers whose AI tool use was limited to learning exercises or personal projects were excluded, as their perspectives would not adequately address questions about organizational accountability and governance. This approach ensured that all participants could provide informed perspectives on ethical responsibility, trust, and governance issues related to AI-generated code. Purposive sampling is widely applied in qualitative software engineering research to obtain information-rich cases that can provide meaningful insights into specific professional practices [13, 18].

Table 1
Participant Profiles Collected via Google Form.

ID	Age Range	Current Role	Years of Experience	Industries Worked In	AI Coding Assistants Used	Frequency of Use per Week (days per week)
P1	20–29	Test Automation Engineer	4–7	Web/Mobile Applications ; Enterprise Software; AI/ML Products	GitHub Copilot; ChatGPT	3–4
P2	20–29	Fullstack Software Developer / Engineer	1–3	Web/Mobile Applications ; Enterprise Software	GitHub Copilot; Cursor; ChatGPT	5–7
P3	20–29	Frontend Developer	1–3	Enterprise Software; AI/ML Products	GitHub Copilot; ChatGPT	Once a week in the past
P4	20–29	Data Analyst / BI Analyst	1–3	Research	ChatGPT	3–4
P5	30–39	Fullstack Software Developer / Engineer	1–3	Web/Mobile Applications ; Enterprise Software; AI/ML Products; Pharmaceuticals; FMCG; Banking; Mining; E-Commerce	Codex (OpenAI)	Less than once per week
P6	40–49	CTO	10+	Enterprise Software	ChatGPT	1–2
P7	20–29	Fullstack Software Developer / Engineer	4–7	Web/Mobile Applications	Cursor; Claude; Copilot	5–7

Seven professionals participated as shown in Table 1, representing roles such as full-stack developers, a frontend developer, a test automation engineer, a data analyst, and a chief technology officer. Participants' professional experience ranged from early-career to senior-level positions. Participants were drawn from sectors including fintech, enterprise software, e-commerce, and consulting domains that vary considerably in governance maturity, security obligations, and IP sensitivity, which was reflected in their reported practices. All reported prior or ongoing use of AI coding assistants, including GitHub Copilot, OpenAI Codex, and ChatGPT. The sample size is consistent with qualitative research suggesting that thematic saturation in focused professional groups often occurs within six to twelve interviews [19]. The goal was analytical rather than statistical generalization [20].

The composition of the sample has interpretive implications worth noting. Six of seven participants fell in the 20 to 29 age range with one to seven years of experience, meaning the findings predominantly reflect early-to-mid career developer perspectives. More experienced developers or engineering managers may hold different accountability perceptions, particularly regarding organizational governance, where seniority often brings both greater policy influence and greater awareness of formal liability structures. The one senior participant (P6, CTO, 10+ years) did offer a distinctly governance-oriented perspective, suggesting that experience level may shape how responsibility is conceptualized. Similarly, participants from fintech and enterprise software contexts operate under stricter regulatory regimes

than those in e-commerce or consulting, which may inflate reported security awareness relative to what would be found in less regulated sectors. These compositional factors do not invalidate the findings but do suggest that the distributed accountability model described here may reflect the perspective of developers who bear practical responsibility without yet holding formal institutional authority, a position that may amplify perceived individual accountability while underweighting organizational and structural factors.

Data collection was conducted in two stages in the autumn of 2025. First, participants completed a short pre-interview survey collecting demographic and professional background information, including role, years of experience, industry exposure, and AI tool usage frequency. This survey served as a contextual profiling instrument and ensured that interview time could focus on substantive ethical issues [21]. Second, semi-structured interviews were conducted remotely via Microsoft Teams. Interviews lasted between 17 and 26 minutes. Semi-structured interviews were selected because they allow consistent thematic coverage while providing flexibility for participants to elaborate on experiences and ethical reasoning [22, 23].

The interview guide was developed following recommended practices for qualitative semi-structured interviews [24] and was informed by themes identified in the literature review. Questions related to responsibility (RQ1) explored developers' perceptions of accountability when AI-generated code introduces vulnerabilities or other forms of harm. Participants were asked who should bear responsibility for problematic AI-generated code and whether the use of AI assistants affects their sense of professional responsibility.

Questions addressing ethical practices (RQ2) examined how developers evaluate and interact with AI-generated suggestions in everyday work. These questions focused on ethical awareness, trust in AI outputs, and review practices related to code quality, security, and intellectual property. For example, participants were asked how they verify the correctness and security of AI-generated code and whether they had encountered situations where they chose not to rely on an AI suggestion. Questions related to future outlook (RQ3) explored transparency, human oversight, and governance mechanisms. Participants were asked what kinds of technical safeguards, transparency features, or organizational policies could support more responsible use of AI coding assistants. The semi-structured format allowed follow-up questions and probing for concrete examples from participants' professional experience. A detailed version of the interview guide is available in the master's thesis on which this study is based [14].

The data were analyzed using reflexive thematic analysis following Braun and Clarke [25], proceeding through six phases: familiarization with the data through repeated transcript readings; generation of initial codes labeling segments related to responsibility, trust, and governance; grouping codes into candidate themes; reviewing themes against the full dataset for coherence; defining and naming themes to capture their analytical substance; and producing the report grounded in verbatim participant statements [26]. The analysis combined deductive elements anticipated from the literature — such as distributed responsibility with inductively emergent patterns, such as the characterization of trust as pragmatic rather than epistemic. An audit trail of codes and theme decisions was maintained to support dependability and confirmability. The researcher's prior experience using AI coding tools may have sensitized attention to productivity themes; this positionality was acknowledged throughout to support reflexivity

Overall, the methodological design enables systematic exploration of how developers interpret accountability, negotiate ethical trade-offs, and envision future safeguards in AI-assisted software development. By grounding the study in practitioners' experiences, the approach provides empirically informed insight into responsibility and governance in the context of AI-generated code.

4. Results

This section presents the findings of the empirical study, drawing on both the contextual survey data and the qualitative interviews. The survey results provide background information about the participants'

professional roles, experience levels, and patterns of AI coding assistant usage. The interview findings form the primary analytical component of the study. The results are organized according to the three research questions. Section 4.1 examines how participants attribute responsibility when ethical or technical problems arise from AI-generated code (RQ1). Section 4.2 explores the ethical considerations that influence developers' evaluation and use of AI coding assistants (RQ2). Section 4.3 presents participants' perspectives on future safeguards, governance mechanisms, and support systems for responsible AI-assisted development (RQ3).

4.1 Responsibility (RQ1)

A central finding across the interviews was the perception that responsibility for AI-generated code is distributed but asymmetrical. All participants agreed that the developer retains immediate responsibility for reviewing, validating, and deploying code, regardless of whether it was written manually or suggested by an AI assistant such as GitHub Copilot or Chat GPT. Developers emphasized that accepting a suggestion constitutes a conscious professional decision. As P1 stated, "If I accept the suggestion, it's my responsibility."

This reflects the alignment with the principles of professional ethics that emphasize the responsibility of the developer for software artifacts [16]. All participants consistently rejected the notion that AI tools could independently bear responsibility, noting that these systems lack agency. However, responsibility was not perceived as exclusive to the developer. Most of the participants (P1-P4, P6 & P7) described a layered model of accountability involving the organization and the AI vendor.

At the organizational level, responsibility was associated with tool adoption decisions, governance policies, and security processes. Developers argued that if a company mandates the use of a coding assistant without providing guidelines or review procedures, it shares accountability for resulting harms. This aligns with the scholarship that emphasizes the role of institutional governance in the deployment of AI [27]. Participant 7 noted that "If management pushes for speed and tells us to cut corners, then the company is more to blame if we ship unsafe AI code."

Vendor responsibility was described as indirect but significant. Participants argued that AI providers should ensure secure model training, provide transparency about limitations, and offer safeguards such as license filtering or vulnerability detection. Although developers did not see vendors as directly liable for specific code deployments, they expected them to minimize systemic risk. This expectation mirrors regulatory discussions surrounding provider obligations in the EU Artificial Intelligence Act [10].

In general, the participants conceptualized responsibility as a shared ecosystem, but with the developer positioned as the final gatekeeper. This finding supports the literature describing human-in-the-loop accountability structures in AI-assisted environments [9]. It is important to distinguish between perceived responsibility and formal structural accountability. Participants reported acting as though they were accountable through code review, validation, and cautious deployment, but who is formally liable when AI generated code causes harm remains unresolved in most of their organizations.

As P1, P2, P4, and P7 stated, 'If I accept the suggestion, it's my responsibility,' reflecting internalized professional accountability. However, organizational structures to support and enforce this accountability were largely absent. This gap represents a governance risk: developers bear the practical burden of responsibility without the institutional support or legal clarity needed to manage systemic AI risks. Closing this gap requires organizational policy that explicitly allocates accountability rather than leaving it to individual discretion. Importantly, no participant reported transferring moral responsibility to the AI system itself. Instead, AI tools were framed as productivity instruments whose outputs must be critically evaluated.

4.2 Ethical Practices (RQ2)

Participants identified three dominant ethical considerations that influence their use of AI-generated code: security, intellectual property (IP), and trust calibration.

Bias and Fairness

Bias and fairness concerns were raised by fewer participants (P5, P7) and were generally framed as more relevant to user-facing AI systems than to backend development logic. This perception is understandable but risks underestimating the scope of bias in code generation, as research shows it manifests in structural backend choices including naming conventions, data schema assumptions, and algorithmic defaults not only in content-facing outputs [15]. P5 noted that if AI assistants are trained on biased datasets, they may reproduce stereotypes or flawed coding practices. This unevenness represents a governance gap: the review practices developers currently rely on are well calibrated to detect security vulnerabilities but poorly suited to identify embedded social biases, which require dedicated audit procedures and organizational awareness rather than individual code review.

This unevenness in ethical awareness likely reflects how bias risk is structured differently from security risk. Security vulnerabilities produce immediate, visible failures such as broken authentication and exploitable inputs that fit naturally into existing review workflows. Bias, by contrast, manifests gradually and systemically: research shows that code generation models trained on historically skewed datasets reproduce structural patterns including biased naming conventions, algorithmic defaults, and data schema assumptions that encode cultural or demographic assumptions without triggering any linting rule or test failure [28, 29]. The absence of bias concern among most participants may therefore reflect not indifference but a genuine absence of tools and vocabulary for recognizing it. This has a direct governance implication: organizations cannot rely on developer vigilance alone to catch bias in AI-generated code, since the same workflow that catches a SQL injection will not catch a discriminatory data assumption. Dedicated bias audits, organizational training, and tool-level interventions such as flagging statistically anomalous naming or schema patterns are needed as a separate governance layer.

Security and Code Quality

Security was the most consistently emphasized concern. The majority of participants (P1–P4, P7) described reviewing AI-generated code with particular scrutiny, especially when dealing with authentication, encryption, or database queries. Several developers (P2, P3, & P7) reported encountering insecure suggestions, such as improper input validation or weak randomization techniques. This aligns with empirical studies demonstrating vulnerability prevalence in AI-generated code [4, 5]. Despite recognizing these risks, participants continued to use AI tools regularly.

Intellectual Property

Intellectual property concerns were frequently mentioned but characterized by uncertainty rather than concrete incidents. Participants were aware of ongoing legal debates regarding AI training data and code similarity [6]. Some reported organizational policies restricting the use of AI-generated code in proprietary projects. Others relied on informal judgment, stating that they avoided copying large AI-generated blocks without modification.

The ambiguity surrounding ownership and licensing influenced cautious usage behavior. Developers acknowledged that they could not easily determine whether a suggestion resembled licensed material. This lack of transparency contributed to risk awareness but did not result in widespread avoidance.

Trust and Professional Judgment

Trust in AI assistants such as OpenAI Codex was described as pragmatic rather than epistemic. Developers trusted outputs when they could quickly understand and verify the logic. When suggestions appeared opaque or overly complex, trust declined. Unexpected outputs reduced confidence and triggered manual validation. This dynamic is consistent with research on algorithm aversion, which suggests that transparency and explainability affect trust calibration [30].

Importantly, participants did not describe AI tools as replacing professional expertise. Instead, they framed them as accelerators that require supervision. Ethical practice was therefore enacted through

review processes rather than abstention from tool use.

4.3 Future Outlook (RQ3)

When asked about improvements that would promote more ethical use of AI coding assistants, participants emphasized three categories: enhanced transparency, stronger technical safeguards, and clearer organizational guidance.

Transparency and Explainability

Participants expressed a desire for clearer indications of confidence levels, training data provenance, or rationale behind suggestions. Although they did not expect full model transparency, they indicated that contextual signals would support better risk assessment. This aligns with calls for explainability and traceability in AI governance frameworks [9]. Several developers (P2, P3, & P7) emphasized that accountability could be improved through governance measures and safeguards, such as careful code review and adherence to organizational policies.

Integrated Security and Compliance Tools

The majority of participants emphasized the importance of safeguards such as code review and testing when using AI-generated code. They highlighted that human oversight, combined with organizational policies, is essential to ensure security and reliability. This suggestion reflects findings in prior literature emphasizing the need for multi-layered safeguards combining AI assistance with automated review mechanisms [5].

Organizational Policies and Training

At the organizational level, participants reported inconsistent governance structures. Most indicated they relied on personal judgment with no formal training or policies in place. As one participant put it: 'We don't have a formal policy; it's really up to us. I'd like some official guidelines.' This absence of institutional frameworks mirrors findings by Mahmud [31], who highlights the difficulty of translating high-level ethical principles into organizational practice.

These cases illustrate that the ethics-practice gap is not uniform but varies significantly with organizational context. Developers emphasized that structured guidance, internal policies, and training on ethical AI use could help clarify responsibility and reduce ambiguity. This recommendation corresponds with literature identifying a gap between high-level ethical frameworks and daily development practice [31]. Notably, participants did not advocate for prohibiting AI tools. Instead, they favored regulated integration supported by human oversight, documentation standards, and shared accountability structures.

4.4 Synthesis of Findings

Across all three research questions, a coherent pattern emerges: developers perceive themselves as primary decision-makers in AI-assisted coding but recognize the necessity of organizational and vendor-level governance. Ethical considerations influence usage primarily through heightened review practices rather than avoidance. Future improvements are envisioned not as constraints but as supportive mechanisms enhancing transparency, security, and clarity of responsibility.

These findings suggest that responsibility in AI-generated code is enacted through professional judgment, contextual trust calibration, and layered governance expectations. Developers neither abdicate responsibility to AI systems nor fully internalize systemic risks alone. Instead, they operate within a distributed accountability model that relies on human oversight, institutional support, and improved technical safeguards.

Collectively, the results provide empirical grounding for ongoing debates about accountability and ethical AI integration in software engineering.

5. Discussion

This study set out to examine how developers perceive responsibility when ethical or technical problems arise from AI-generated code (RQ1), what ethical considerations shape their evaluation and use of such code (RQ2), and what future safeguards they consider necessary (RQ3). The findings reveal a consistent pattern of distributed but developer-centered accountability, conditional trust, and a demand for clearer governance mechanisms.

With respect to responsibility (RQ1), participants overwhelmingly positioned themselves as the final gatekeepers of AI-generated code. Even when using tools such as GitHub Copilot or ChatGPT, developers emphasized that accepting and committing code constitutes a deliberate professional act. This aligns with established software engineering ethics frameworks, including the Association for Computing Machinery Code of Ethics, which assigns responsibility to practitioners for the safety and integrity of their artifacts[16]. At the same time, responsibility was not viewed as purely individual. Participants articulated a layered accountability model in which organizations share responsibility through governance structures and vendors bear obligations related to model quality, transparency, and risk mitigation. This distributed perspective resonates with contemporary AI governance scholarship emphasizing shared responsibility across developers, deployers, and providers [27, 9].

Regarding ethical practices (RQ2), security emerged as the dominant concern influencing developers' evaluation of AI-generated code. Participants described heightened scrutiny when reviewing suggestions, particularly in security-sensitive contexts. This reflects empirical findings demonstrating vulnerability prevalence in AI-generated outputs [4, 5]. Yet, consistent with the "trust paradox" identified by Kreps et al. [7], developers continued to rely on AI tools despite recognizing their fallibility. Rather than rejecting AI assistance, participants calibrated trust contextually, using it for boilerplate or low-risk tasks while manually validating complex logic. Intellectual property concerns were also present but characterized by uncertainty, reflecting ongoing legal ambiguity [6]. Bias was mentioned less frequently, suggesting that ethical awareness may be uneven across different risk domains.

The conditional, context-dependent trust described here mirrors patterns observed in other high-stakes AI domains. In clinical decision support, studies of experienced practitioners show that continued reliance on AI recommendations persists even after observing errors, with heightened scrutiny applied selectively to high-risk cases while AI assistance is accepted more readily for routine ones [32], a dynamic structurally parallel to developers trusting AI for boilerplate while manually reviewing authentication logic. Research on AI use in military and national security decision-making similarly finds that expert operators balance selective distrust against workflow efficiency rather than adopting uniform skepticism or uniform reliance [33]. What distinguishes the software development context is the relative legibility of the artifact, since code can in principle be read and verified in ways that a medical diagnosis or threat assessment cannot. The persistence of the trust paradox even where verification is feasible suggests that the mechanism is not primarily epistemic but pragmatic: developers trust AI tools because the cost of full verification exceeds the perceived risk of partial reliance, not because they lack the capacity to evaluate outputs.

Concerning the future outlook (RQ3), participants favored regulated integration rather than restriction. Desired improvements centered on transparency features, integrated security scanning, and clearer organizational guidelines. These expectations align with normative frameworks such as the UNESCO Recommendation on the Ethics of Artificial Intelligence, which emphasizes human oversight, explainability, and accountability [9]. The findings suggest that developers are willing to assume responsibility but require institutional and technical support to do so effectively.

Overall, the discussion highlights a critical tension: while developers remain the primary moral agents in AI-assisted coding, the risks introduced by generative systems exceed the scope of purely individual control. Effective governance therefore requires coordinated accountability structures that combine professional judgment, organizational policy, and vendor-level safeguards. By grounding abstract regulatory debates in practitioner perspectives, this study contributes empirical insight into how responsibility is interpreted and operationalized in contemporary AI-supported software development.

Positioning these findings against prior literature reveals both convergence and tension. The dis-

tributed accountability model aligns with Floridi and Cowls' framework [27], but unlike its balanced view of responsibility, participants placed most practical accountability on individual developers, while organizational and vendor responsibility remained limited in practice. This suggests governance theory has advanced faster than organizational implementation. The trust paradox finding also extends Kreps et al. [7]: even expert developers who can verify AI outputs still display conditional reliance, indicating that the paradox stems less from opacity and more from productivity pressures and workflow integration. Similarly, while Mahmud [31] identifies an ethics-practice gap at the organizational level, this study locates it at the intersection of individual judgment and weak institutional support.

A further contribution is identifying how the gap between ethical frameworks and practice might be concretely bridged. Participants proposed specific mechanisms confidence scoring surfaced in the IDE, license-flagging for restricted code, mandatory AI disclosure in commits, and static analysis tools coupled to the suggestion pipeline that translate abstract governance principles into operational toolchain features. Importantly, the findings on bias suggest that closing this gap requires different mechanisms than those sufficient for security: bias risks cannot be detected through standard code review and instead require dedicated audit procedures and training programs that build developer capacity to recognize structural bias in generated code.

6. Conclusions

This study examined how developers perceive responsibility when ethical or technical problems arise from AI-generated code (RQ1), which ethical considerations influence their evaluation and use of such tools (RQ2), and what future mechanisms could strengthen ethical integration (RQ3).

The findings indicate that developers retain primary responsibility for AI-generated code, even when using assistants such as GitHub Copilot and ChatGPT. Responsibility is understood as distributed across an ecosystem including organizations and AI vendors, but remains operationally centered on the developer as the final decision-maker. This aligns with professional accountability principles articulated by the Association for Computing Machinery Code of Ethics and broader AI governance frameworks [16, 9, 10].

Security concerns, intellectual property uncertainty, and calibrated trust practices shape how developers evaluate AI-generated code. Rather than rejecting these tools, participants described selective reliance combined with intensified review. This suggests that ethical practice in AI-assisted development is enacted through human oversight rather than technological abstention.

Looking forward, developers advocate for enhanced transparency, integrated security safeguards, and clearer organizational guidelines. These findings reinforce the need for coordinated governance structures that support individual accountability while addressing systemic risks. By grounding normative debates about AI responsibility in practitioner experience, this study contributes empirical insight into how accountability is interpreted and operationalized in contemporary software engineering.

Several limitations should inform the interpretation of these findings. The sample of seven participants restricts generalizability across industries and cultural contexts. All data are self reported, which may introduce social desirability bias regarding review practices. Future research should employ larger and mixed-method designs, combining interview data with observational or log-based evidence of actual development behavior, and longitudinal approaches to track how responsibility perceptions evolve as AI tools mature.

Declaration of Generative AI

During the preparation of this paper, the author used AI-assisted tools for language and structural support. A large language model (ChatGPT) was used to assist with improving grammar, sentence structure, spelling, and the identification of typographical errors throughout the manuscript. All AI-assisted suggestions were carefully reviewed and edited as necessary. The author assumes full responsibility for the accuracy, integrity, and originality of the final submitted work.

References

[1] S. Peng, E. Kalliamvakou, P. Cihon, M. Demirer, The impact of ai on developer productivity: Evidence from github copilot, 2023. URL: https://arxiv.org/abs/2302.06590. arXiv:2302.06590.

[2] T. Dohmke, M. Iansiti, G. Richards, Sea change in software development: Economic and productivity analysis of the ai-powered developer lifecycle, 2023. URL: https://arxiv.org/abs/2306.15033. arXiv:2306.15033.

[3] S. Overflow, Ai | 2025 stack overflow developer survey, 2025. URL: https://survey.stackoverflow.co/2025/ai.

[4] Y. Fu, P. Liang, A. Tahir, Z. Li, M. Shahin, J. Yu, J. Chen, Security weaknesses of copilot-generated code in github projects: An empirical study, ACM Trans. Softw. Eng. Methodol. 34 (2025). URL: https://doi.org/10.1145/3716848. doi:10.1145/3716848.

[5] Veracode, Genai code security report: Assessing the security of using llms for coding, 2025. URL: https://www.veracode.com/wp-content/uploads/2025_GenAI_Code_Security_Report_Final.pdf, accessed: 2026-03-03.

[6] M. Germanò, Do as i say, not as i code: Github's copilot prompts ip litigation with international implications, New York University Journal of International Law and Politics 57 (2025). URL: https://nyujilp.org/wp-content/uploads/2025/03/Annotation_Germano_Macroed.pdf, accessed: 2026-03-03.

[7] S. Kreps, J. George, P. Lushenko, A. Rao, Exploring the artificial intelligence "trust paradox": Evidence from a survey experiment in the united states, PLOS ONE 18 (2023) 1–21. URL: https://doi.org/10.1371/journal.pone.0288109. doi:10.1371/journal.pone.0288109.

[8] S. Baltes, T. Speith, B. Chiteri, S. Mohsenimofidi, S. Chakraborty, D. Buschek, On the Need to Rethink Trust in AI Assistants for Software Development: A Critical Review , IEEE Transactions on Software Engineering (5555) 1–18. URL: https://doi.ieeecomputersociety.org/10.1109/TSE.2026.3659804. doi:10.1109/TSE.2026.3659804, accessed: 2026-03-03.

[9] UNESCO, Recommendation on the ethics of artificial intelligence, 2022. URL: https://www.unesco.org/en/artificial-intelligence/recommendation-ethics, accessed: 2026-03-03.

[10] E. Parliament, Council, Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act), Official Journal L 2024/1689 (2024), 2024. URL: https://eurlex.europa.eu/eli/reg/2024/1689/oj/eng, accessed: 2026-03-03.

[11] R. A. Stebbins, Exploratory Research in the Social Sciences, Qualitative Research Methods, SAGE Publications, Inc., 2001. URL: https://doi.org/10.4135/9781412984249. doi:10.4135/9781412984249.

[12] U. Flick, An Introduction to Qualitative Research, 7th ed., SAGE Publications Ltd, 2023. URL: https://uk.sagepub.com/en-gb/eur/an-introduction-to-qualitative-research/book278983.

[13] C. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Transactions on Software Engineering 25 (1999) 557–572. doi:10.1109/32.799955.

[14] A. S. Wijaya, Exploring Ethical Awareness and Practices in Software Development: Study of AI Coding Assistant Tools Usage Among Developers, Master's thesis, Tampere University, 2025. URL: https://urn.fi/URN:NBN:fi:tuni-2025113011082.

[15] Y. Liu, X. Chen, Y. Gao, Z. Su, F. Zhang, D. Zan, J.-G. Lou, P.-Y. Chen, T.-Y. Ho, Uncovering and quantifying social biases in code generation, NIPS '23, Curran Associates Inc., Red Hook, NY, USA, 2023. URL: https://dl.acm.org/doi/10.5555/3666122.3666232.

[16] Association for Computing Machinery, Acm code of ethics and professional conduct, https://www.acm.org/code-of-ethics/software-engineering-code, 2018. Accessed: 2026-03-03.

[17] Institute of Electrical and Electronics Engineers, Ieee 7000-2021 – ieee standard model process for addressing ethical concerns during system design, https://standards.ieee.org/ieee/7000/6781/, 2021. Accessed: 2026-03-03.

[18] L. A. Palinkas, S. M. Horwitz, C. A. Green, J. P. Wisdom, N. Duan, K. Hoagwood, Purposeful sampling for qualitative data collection and analysis in mixed method implementation research, Administration and Policy in Mental Health and Mental Health Services Research 42 (2015) 533–544. doi:10.1007/s10488-013-0528-y.

[19] G. Guest, A. Bunce, L. Johnson, How many interviews are enough?: An experiment with data saturation and variability, Field Methods 18 (2006) 59–82. doi:10.1177/1525822X05279903.

[20] R. K. Yin, Case Study Research and Applications: Design and Methods, 6th ed., SAGE Publications, Inc., 2017. Available: https://us.sagepub.com/en-us/nam/case-study-research-and-applications/book250150.

[21] A. Fink, How to Conduct Surveys: A Step-by-Step Guide, 6th ed., SAGE Publications Ltd, 2016. Available: https://eric.ed.gov/?id=ED565650.

[22] S. Brinkmann, S. Kvale, InterViews: Learning the Craft of Qualitative Research Interviewing, 3rd ed., SAGE Publications, Inc., 2014. Available: https://uk.sagepub.com/en-gb/eur/interviews/book239402.

[23] H. Rubin, I. Rubin, Qualitative Interviewing: The Art of Hearing Data, 3rd ed., SAGE Publications, Thousand Oaks, 2012. Available: https://www.scirp.org/reference/referencespapers?referenceid=1770556.

[24] H. Kallio, A.-M. Pietilä, M. Johnson, M. Kangasniemi, Systematic methodological review: developing a framework for a qualitative semi-structured interview guide, Journal of Advanced Nursing 72 (2016) 2954–2965. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jan.13031. doi:https://doi.org/10.1111/jan.13031. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/jan.13031.

[25] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative Research in Psychology 3 (2008) 77–101. URL: https://doi.org/10.1191/1478088706qp063oa. doi:10.1191/1478088706qp063oa.

[26] L. S. Nowell, J. M. Norris, N. J. Moules, Thematic analysis: Striving to meet the trustworthiness criteria, International Journal of Qualitative Methods 16 (2017) 1609406917733847. URL: https://doi.org/10.1177/1609406917733847. doi:10.1177/1609406917733847.

[27] L. Floridi, J. Cowls, A Unified Framework of Five Principles for AI in Society, Harvard Data Science Review 1 (2019). URL: https://doi.org/10.1162/99608f92.8cd550d1.

[28] R. Krasniqi, D. Xu, M. Vieira, Se perspective on llms: Biases in code generation, code interpretability, and code security risks, ACM Computing Surveys 58 (2025). URL: https://doi.org/10.1145/3774324. doi:10.1145/3774324.

[29] D. Huang, J. M. Zhang, Q. Bu, X. Xie, J. Chen, H. Cui, Bias testing and mitigation in llm-based code generation, ACM Transactions on Software Engineering and Methodology 35 (2025). URL: https://doi.org/10.1145/3724117. doi:10.1145/3724117.

[30] B. J. Dietvorst, J. P. Simmons, C. Massey, Algorithm aversion: People erroneously avoid algorithms after seeing them err, Journal of Experimental Psychology: General 144 (2015) 114–126. doi:10.1037/xge0000033.

[31] H. Mahmud, Ethical implications of ai-assisted coding in software engineering, 2025. URL: https://urn.fi/URN:NBN:fi:amk-2025061122482, all rights reserved. Commercial use is prohibited.

[32] A. Küper, G. C. Lodde, E. Livingstone, D. Schadendorf, N. Krämer, Psychological factors influencing appropriate reliance on ai-enabled clinical decision support systems: Experimental web-based study among dermatologists, Journal of Medical Internet Research 27 (2025) e58660. URL: https://www.jmir.org/2025/1/e58660. doi:10.2196/58660.

[33] P. Lushenko, Ai, trust, and the war room: Evidence from a conjoint experiment in the us military, Cambridge Forum on AI Law and Governance 1 (2026) 1–24. URL: https://doi.org/10.1017/cfl.2025.10019. doi:10.1017/cfl.2025.10019, licensed under CC BY-NC-SA 4.0.