Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/NA11.20260197
Published: June 9, 2026

The Importance of Human Critical Thinking in the Validation of AI-Generated Code in Software Development Projects: A Systematic Literature Review

Leandro Fernandes de Oliveira
Independent Researcher
Calgary, Alberta, Canada
leandro.oliveira@live.com

Abstract

Artificial Intelligence (AI) has been widely adopted to automate tasks in software development, providing support for code generation, testing, and maintenance. This paper presents a systematic literature review on the role of human critical thinking in the validation of AI-generated code, highlighting the benefits, challenges, and limitations of automation in software engineering workflows. The review identifies that, while AI increases efficiency, reduces errors, and optimizes processes, human oversight remains essential to ensuring quality, ethics, and security in software development. Results indicate that the integration of AI tools with critical human evaluation is fundamental to improving system reliability, particularly in high-stakes domains such as financial technology and safety-critical systems.

Keywords
Artificial intelligence, software development, automation, critical thinking, code validation, AI-generated code, software quality, human-AI collaboration

1. Introduction
The use of Artificial Intelligence (AI) in software development has expanded rapidly, offering automation of tasks, process optimization, and decision-making support. AI-powered tools such as GitHub Copilot, Visual Studio IntelliCode, and AutoML platforms enable automatic code generation, bug identification, and test optimization, significantly accelerating development cycles (GitHub 2023).

Despite these benefits, AI-generated code is not fully reliable and may contain errors, security vulnerabilities, or algorithmic biases. In this context, human critical thinking becomes essential to validate, correct, and improve the outputs produced by automated systems (Russell and Norvig 2021). This concern is particularly relevant in regulated industries such as financial technology, healthcare, and critical infrastructure, where software defects can have significant consequences for end users and regulatory compliance.

This study conducts a systematic literature review on the application of AI in software development, addressing ethical and technical challenges, innovation opportunities, and the necessity of human supervision. The primary objective is to analyze how the integration of AI tools with critical human evaluation can increase the efficiency and reliability of software development, providing actionable insights for practitioners and researchers in the field.

2. Theoretical Background
2.1 Human Intelligence and Critical Thinking
Human intelligence is defined as the capacity to understand, learn, and apply knowledge across different contexts. Vygotsky (1978) positions language as a mediator between the individual and the environment, shaping intelligence

through social and historical processes. Resnick (1976) and Almeida (1994) reinforce that intelligence encompasses cognitive processes, strategies, and experiences applied to complex problem-solving. In the context of software engineering, critical thinking enables practitioners to evaluate AI outputs beyond syntactic correctness, assessing semantic validity, security implications, and alignment with system requirements.

2.2 Artificial Intelligence in Software Engineering

Artificial Intelligence is a branch of computer science focused on creating systems capable of simulating aspects of human intelligence, including pattern recognition, learning, decision-making, and adaptation (McCarthy et al. 2006). Techniques such as machine learning, artificial neural networks, and natural language processing have been applied in software development to automate tasks, optimize processes, and identify patterns in large datasets (Domingos 2015).

The emergence of large language model (LLM)-based tools such as GitHub Copilot has further intensified this dynamic, as these systems can generate plausible-looking code that contains subtle logical errors or security vulnerabilities not immediately apparent to less experienced reviewers (Chen et al. 2021). Human intervention remains necessary for validation and correction of such outputs (Barenkamp et al. 2020).

2.3 Software Development Lifecycle and Quality Standards

Software development involves requirements analysis, design, coding, testing, and maintenance. The adoption of best practices, coding standards, and agile frameworks aims to improve quality, productivity, and maintainability (Martin 2008). International quality standards such as ISO/IEC 25010 (ISO/IEC 25010 2011) and IEEE guidelines (IEEE Std 730 2014) establish frameworks for evaluating software quality attributes including functional suitability, reliability, security, and maintainability. Human critical thinking remains essential within these frameworks to validate the reliability of generated code and prevent critical errors that automated tools may fail to detect.

3. Methodology

3.1 Research Objective

The objective of this study is to analyze, through a systematic literature review, the impact of AI on software development, highlighting the importance of human critical thinking in the validation of automatically generated code.

3.2 Literature Selection Criteria

Articles, conference papers, and books published between 2000 and 2024 were selected, prioritizing academic sources, case studies, and reference reports in software development and AI. Databases used include IEEE Xplore, ACM Digital Library, and SpringerLink. Selection criteria required relevance to at least one of the following themes: AI-assisted code generation, human oversight in automated systems, software quality assurance, or critical thinking in engineering contexts.

The review followed a structured protocol aligned with the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework, adapted to the scope and topic boundaries of software engineering and AI-assisted development. Search queries combined controlled terms across the three databases listed above. Representative search strings included: (“artificial intelligence” OR “machine learning” OR “large language models”) AND (“code generation” OR “AI-assisted development” OR “automated software engineering”); (“human oversight” OR “human-in-the-loop” OR “critical thinking”) AND (“software validation” OR “code review” OR “software quality”); and (“GitHub Copilot” OR “IntelliCode” OR “Codex”) AND (“validation” OR “reliability” OR “vulnerability”). Reference lists of the most relevant studies were also screened to identify foundational sources, including books and international standards that contextualise the theoretical framework of the review.
Inclusion and exclusion criteria are summarised in Table 1, and were applied first at the title-and-abstract screening stage and then at the full-text review stage.

Table 1. Inclusion and exclusion criteria applied during study selection

Dimension	Inclusion criteria	Exclusion criteria
Topic relevance	AI-assisted code generation; human oversight in automated systems; software quality assurance; critical thinking in engineering contexts	AI applications outside software engineering (e.g., image recognition, robotics, autonomous vehicles)
Source type	Peer-reviewed journal articles; conference proceedings; technical books; international standards (ISO/IEC, IEEE); widely-cited industry technical reports	Editorials, opinion pieces without methodological grounding, blog posts, marketing materials
Publication period	2000–2024, with selected foundational works on intelligence and learning theory included for theoretical grounding	Pre-2000 publications without sustained influence on the field
Engineering focus	Studies addressing validation, reliability, security, or quality of AI-generated artifacts; studies on human oversight in AI-assisted workflows	Studies focused solely on tooling features without reference to validation, oversight, or quality outcomes

Source: Author's elaboration based on review protocol design (2025)

The study selection process followed four stages consistent with PRISMA: (i) identification of records through database searching using the strings described above; (ii) screening of titles and abstracts against the inclusion criteria; (iii) full-text assessment of records that passed screening; and (iv) final inclusion of studies retained for qualitative synthesis. Records were excluded at the screening stage when they addressed AI in domains unrelated to software engineering, when AI was discussed only at a conceptual level without engineering implications, or when the work focused solely on tooling features without reference to validation or oversight. The final corpus retained for qualitative synthesis is reflected in the references cited throughout this paper, and includes peer-reviewed studies, technical books, and international standards selected for their direct relevance to the research objective.

3.3 Tools and Techniques Analyzed

The review considered the following AI tools and frameworks applied to software development:

- Machine Learning platforms: Scikit-Learn, Keras, TensorFlow
- Natural Language Processing (NLP): techniques for code analysis and generation
- AI-assisted development platforms: GitHub Copilot, Visual Studio IntelliCode, Amazon CodeWhisperer
- AutoML frameworks for test generation and defect prediction

3.4 Data Collection and Analysis

Data were categorized into four dimensions: (1) automation of repetitive tasks; (2) error detection and correction; (3) performance optimization; and (4) limitations and ethical challenges. The analysis followed a thematic synthesis approach, in which findings extracted from each included study were coded against these four dimensions and then aggregated to identify patterns, trends, and gaps in the literature, with particular emphasis on evidence for or against the sufficiency of automated validation without human oversight. The synthesis prioritised consistency of findings across multiple sources and explicitly documented cases where the literature converges and where it remains underdeveloped.

4. Results

The literature analysis identified the primary impacts of AI application in software development. Results are summarized in Table 2.

Table 2. Summary of AI impacts in software development

Category	Identified Benefit	Metric / Impact
Task Automation	Reduction in testing and debugging time	Up to 40% reduction in test cycle time (Sorte et al. 2015)
Error Detection	More precise bug identification	Up to 30% more efficient than traditional methods
Performance Optimization	Improvements in software performance	Reduction in failures and increased reliability
User Experience	Personalization and software quality	25% increase in user satisfaction in field tests (Khan et al. 2023)

Source: Author's elaboration based on literature review (2025)

In addition to the benefits identified, the literature consistently reports limitations including dependency on training data quality, potential biases in algorithms, and the necessity of human supervision for final code validation. Studies in regulated domains — particularly financial systems and healthcare — report that automated validation alone is insufficient to meet compliance requirements established by standards such as ISO/IEC 25010 (ISO/IEC 25010 2011).

4.1 Research Gaps Identified

Synthesising the reviewed evidence, four research gaps emerged from the analysis. These gaps motivate the discussion in Section 5 and the directions for future work proposed in Section 6.

Gap 1 — Absence of quantitative measures for the combined effectiveness of AI and human review. The reviewed literature suggests that AI-assisted development can deliver efficiency gains (Sorte et al. 2015), but does not establish standardised metrics that capture the joint contribution of automated suggestions and subsequent human validation. Without such metrics, organisations cannot compare review protocols or attribute reliability outcomes to specific stages of the workflow.

Gap 2 — Limited empirical evidence in regulated and safety-critical domains. The reviewed literature broadly acknowledges that automated validation alone is insufficient to meet compliance requirements in regulated domains such as financial technology, healthcare, and critical infrastructure (ISO/IEC 25010 2011), yet sector-specific empirical studies that connect AI-assisted development practices to measurable compliance outcomes (e.g., ISO/IEC 25010, IEEE 730) remain scarce.

Gap 3 — Insufficient treatment of accountability and transparency for AI-generated artifacts. Ethical concerns about responsibility for automated decisions and the opacity of model behaviour are repeatedly raised (Russell and Norvig 2021; Chen et al. 2021), but the literature offers limited guidance on how engineering teams should record, attribute, and audit decisions taken on AI-suggested code during the development lifecycle.

Gap 4 — Lack of structured protocols for integrating critical thinking into AI-assisted workflows. Although the necessity of human oversight is widely acknowledged (Barenkamp et al. 2020; Korzeniowski and Goczyla 2019), the reviewed literature does not provide reproducible protocols or maturity models that specify when and how human review should be inserted across requirements, design, coding, testing, and maintenance stages of the software development lifecycle.

5. Discussion

The results demonstrate that AI has significant potential to automate repetitive tasks, reduce errors, and optimize development processes. Tools such as GitHub Copilot and Visual Studio IntelliCode exemplify systems that increase productivity but still require critical human intervention to ensure code reliability, prevent propagation of vulnerabilities or biases, and apply quality standards and programming best practices (Korzeniowski and Goczyla 2019).

While AI accelerates the development cycle, the literature consistently indicates that no tool fully replaces human analysis, especially in critical contexts such as financial systems, healthcare, or high-security infrastructure (Ng 2016). Organizations adopting AI-assisted development without structured human review protocols experience higher rates of security incidents and regulatory findings compared to those that maintain human oversight as a mandatory step in the development lifecycle (Stripe 2018).

Ethical challenges were also identified across the reviewed literature, including accountability for automated decisions, transparency in AI processes, and workforce impact. Future research should explore quantitative methodologies that measure the combined effectiveness of AI and human critical thinking, providing evidence-based guidelines for optimizing the balance between automation efficiency and human oversight quality.

6. Conclusion

This systematic literature review demonstrates that Artificial Intelligence can significantly improve software development by promoting automation, error detection, and performance optimization. However, human critical thinking remains essential to validate, correct, and improve automatically generated code, ensuring quality, ethics, and security across the software development lifecycle.

As AI adoption accelerates across industries, establishing structured human review protocols becomes a strategic imperative rather than an optional practice. Organizations that treat human critical thinking as a core component of AI-assisted workflows are better positioned to deliver reliable, secure, and compliant software at scale. Future research should investigate measurable evaluation methods combining AI performance metrics with human review indicators to provide a more robust analysis of the impact of automation on the software development cycle.

References

Almeida, L. S., *A inteligência humana: uma perspectiva psicológica*, Porto Editora, 1994.

Barenkamp, M., Rebstadt, J. and Thomas, O., Applications of AI in classical software engineering, *AI Perspectives*, vol. 2, no. 1, pp. 1–10, 2020.

Chen, M. et al., Evaluating large language models trained on code, *arXiv preprint* arXiv:2107.03374, 2021.

Domingos, P., *The Master Algorithm*, Basic Books, 2015.

GitHub, GitHub Copilot: Your AI pair programmer, Available: https://github.com/features/copilot, 2023.

IEEE Std 730, *IEEE Standard for Software Quality Assurance Processes*, IEEE, 2014.

ISO/IEC 25010, Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE), ISO/IEC, 2011.

Khan, A. R., Fatima, J. and Ahmed, S., Crossover of artificial intelligence and software development lifecycle, *International Journal of Computing and Related Technologies*, vol. 4, no. 1, pp. 35–42, 2023.

Korzeniowski, L. and Goczyla, K., Artificial intelligence for software development: The present and the challenges for the future, *Biuletyn Wojskowej Akademii Technicznej*, vol. 66, no. 1, pp. 15–32, 2019.

Martin, R. C., *Clean Code: A Handbook of Agile Software Craftsmanship*, Prentice Hall PTR, 2008.

McCarthy, J., Minsky, M. L., Rochester, N. and Shannon, C. E., A proposal for the Dartmouth summer research project on artificial intelligence, *AI Magazine*, vol. 27, no. 4, pp. 12–14, 2006.

Ng, A., *Machine Learning Yearning*, DeepLearning.AI, Available: https://www.deeplearning.ai/machine-learning-yearning/, 2016.

Resnick, L. B., *The Nature of Intelligence*, Lawrence Erlbaum Associates, 1976.

Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, 4th Edition, Prentice Hall, 2021.

Sorte, B. W., Joshi, P. P. and Jagtap, V., Use of artificial intelligence in software development life cycle, *International Journal of Advanced Engineering and Global Technology*, vol. 3, no. 3, pp. 398–403, 2015.

Stripe, The Developer Coefficient, Available: https://stripe.com/files/reports/the-developer-coefficient.pdf, 2018.

Vygotsky, L. S., *Mind in Society: The Development of Higher Psychological Processes*, Harvard University Press, 1978.

Biography

Leandro Fernandes de Oliveira is an Independent Researcher based in Calgary, Alberta, Canada, with more than fourteen years of professional experience in software engineering, mobile application development, and production-grade financial technology systems. His research interests include artificial intelligence in software engineering, regulatory compliance frameworks for cryptocurrency trading platforms, server-driven user interface architectures,

and human–AI collaboration in code validation workflows. Mr. Oliveira is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE Membership #100993580). He serves as a peer reviewer for several international venues, including the 2026 ACM Practice and Experience in Advanced Research Computing conference (PEARC'26), where he is a member of the Technical Program Committee for the Panels and Birds-of-a-Feather track. He has also reviewed for the IEEE TPC for the 16th International Conference on Computational Intelligence and Communication Networks, and contributes as a peer reviewer for the Industrial Engineering and Operations Management Society (IEOM). He is also registered as a reviewer with the Association for Computing Machinery (ACM) through the ScholarOne manuscript management platform. His current research focuses on the intersection of AI-assisted development, software quality assurance in regulated domains, and the integration of structured human oversight into production engineering workflows. He can be reached at leandro.oliveira@live.com and publishes under ORCID identifier 0009-0003-6560-418X.