

# Preserving Competitional Power in the Age of AI: A Case for Hybrid Programming Literacy in University Curricula

Simon Poon

18-8-2025

## Abstract

As artificial intelligence continues to transform software development, the role of human developers is evolving rather than diminishing. This paper explores the dynamics of human-AI collaboration, emphasizing the need for hybrid literacy among developers. It argues that while AI tools enhance productivity and automate routine tasks, human oversight, creativity, and ethical reasoning remain indispensable. A newly introduced section highlights the enduring importance of human-to-human interaction, especially in team-based and user-centric projects. Emotional intelligence, interpersonal communication, and collaborative spirit are shown to be vital complements to technical expertise. The paper concludes that the future of software development lies not in replacing humans, but in fostering strategic alliances between intelligent systems and emotionally intelligent teams.

## 1 Introduction

The integration of AI into software development has sparked both innovation and uncertainty. Tools like CodeRabbit and GitHub Copilot promise rapid code generation and intelligent assistance, leading some to speculate that traditional programming skills may soon be obsolete. However, this assumption overlooks the limitations of AI reasoning, the complexity of software systems, and the ethical stakes of automation. In domains such as healthcare, finance, and autonomous systems, human oversight is not optional—it is foundational. This paper introduces the term *competitional power* to describe the strategic advantage held by individuals who combine traditional programming expertise with AI fluency. Competitional power can be understood as a multidimensional capability encompassing technical fluency, epistemic awareness, ethical judgment, and strategic adaptability. It reflects a developer’s ability to remain relevant and effective in AI-augmented environments—not merely by using tools, but by shaping their application. We argue that universities must urgently reform their curricula to cultivate this hybrid literacy, equipping graduates to navigate—and shape—the future of software engineering. [5] [10]

## 2 AI Programming vs. Traditional Programming

### 2.1 Misconceptions of Replacement

AI programming tools do not “think” in the way human developers do. CodeRabbit, for example, generates code based on pattern recognition from large datasets, but lacks contextual understanding of system architecture or business logic [1]. While it can produce syntactically correct code, it often requires human intervention for debugging and validation. [4]

### 2.2 Example: CodeRabbit in Practice

Consider a scenario where CodeRabbit generates a REST API for a financial application. The code may compile, but it might:

- Fail to handle edge cases (e.g., null inputs, race conditions)
- Omit security protocols (e.g., input sanitization, authentication)
- Misalign with business rules (e.g., transaction limits)

A developer must manually test and debug these issues—tasks that require domain knowledge and critical thinking. [3]

## 3 The Role of Human Intervention

### 3.1 Debugging and Testing

AI-generated code often lacks robustness. Studies show that Copilot-generated code contains security vulnerabilities in up to 40% of cases [2]. However, more recent analyses from 2024 suggest a modest improvement, with rates closer to 36% though the underlying risks of overtrust and insufficient validation persist. Human oversight is essential to:

- Write unit and integration tests
- Validate performance under load
- Ensure compliance with industry standards

## 3.2 Systematic Thinking

AI tools operate at the micro level (functions, snippets), but software engineering demands macro-level reasoning:

- Modular design
- Dependency management
- Version control
- Ethical considerations

These cannot be outsourced to AI without risking system failure or ethical breaches. [7]

## 4 AI in Requirements Elicitation

Requirements elicitation is a critical phase in software development, where misunderstandings or omissions can lead to costly rework or project failure. AI tools are increasingly being used to assist in this phase, but their effectiveness depends heavily on human guidance and contextual awareness. [6]

### 4.1 Traditional vs. AI-Enhanced Elicitation

Traditionally, requirements are gathered through stakeholder interviews, questionnaires, document analysis, and workshops. These methods rely on human intuition, negotiation, and domain expertise. AI-enhanced elicitation introduces tools that can:

- Extract requirements from unstructured data (emails, chat logs, meeting transcripts)
- Generate initial use cases or user stories from minimal input
- Cluster and prioritize feedback using natural language processing (NLP)

For example, enterprise tools like Copilot4DevOps—used in Azure DevOps environments—can parse stakeholder conversations and suggest functional requirements. RequAI, while still emerging and less widely adopted, represents a conceptual direction for NLP-assisted elicitation. However, these suggestions often lack nuance and must be validated by domain experts.

## 4.2 Stakeholder Communication and Semantic Mapping

AI can assist in bridging communication gaps between technical teams and non-technical stakeholders by:

- Translating business language into technical specifications
- Identifying semantic overlaps and contradictions in stakeholder input
- Visualizing requirement dependencies using graph-based models

Yet, AI struggles with implicit requirements—those not stated but assumed—and with interpreting emotional or political subtext in stakeholder priorities. Human facilitators remain essential to navigate these subtleties.

## 4.3 Ambiguity Resolution and Conflict Detection

AI tools can flag ambiguous phrases like “fast response” or “secure access,” but they cannot resolve them without human input. Techniques such as:

- Prompting stakeholders for clarification
- Offering multiple interpretations for review
- Suggesting industry-standard definitions

must be guided by developers or analysts. AI can assist, but not arbitrate. [8]

## 4.4 Bias Detection and Ethical Oversight

AI can help detect bias in requirements—such as exclusionary language or assumptions that disadvantage certain user groups. For instance:

- Identifying gendered language in user personas
- Flagging accessibility oversights
- Highlighting cultural assumptions in UI design

However, AI itself may carry biases from its training data. Human oversight is required to ensure ethical integrity and inclusivity.

## 4.5 Participatory Design and Co-Creation

Modern elicitation increasingly involves participatory design, where users co-create features with developers. AI can facilitate this by:

- Generating mockups or prototypes from verbal descriptions
- Simulating user flows based on behavioral data
- Offering real-time feedback during design workshops

But AI cannot replace empathy, negotiation, or the creative tension that drives innovation. These are inherently human processes.

## 4.6 Limitations and Risks

Despite its promise, AI in requirements elicitation faces several risks:

- Overfitting to historical data: AI may suggest outdated or irrelevant features.
- Hallucination: Language models may invent requirements not grounded in stakeholder input.
- Loss of stakeholder trust: Over-reliance on AI may alienate users who feel unheard or misrepresented.

These risks underscore the need for a human-in-the-loop approach, where AI augments but does not replace human judgment.

# 5 Curriculum Reform: A Hybrid Literacy Model

The rapid integration of AI into software development demands a rethinking of how universities prepare students for the workforce. Traditional programming education—focused on syntax, algorithms, and manual debugging—no longer suffices. At the same time, over-reliance on AI tools without foundational understanding risks producing developers who lack critical judgment. A Hybrid Literacy Model bridges this gap by combining traditional software engineering principles with AI fluency, ethical reasoning, and epistemic awareness. [9]

## 5.1 Defining Hybrid Literacy

Hybrid Literacy refers to the ability to:

- Understand and apply core programming concepts (e.g., data structures, control flow, software architecture)
- Use AI tools like CodeRabbit, GitHub Copilot, and Replit Ghostwriter effectively and responsibly
- Critically evaluate AI-generated code for correctness, security, and ethical implications
- Recognize the epistemic boundaries of AI reasoning and avoid overtrust

This literacy is not merely technical—it’s cognitive, ethical, and strategic.

### Assessing Competitional Power

Competitional power may be evaluated through a combination of core dimensions:

- **AI Tool Fluency:** Ability to prompt, interpret, and refine AI-generated code effectively.
- **Critical Oversight:** Capacity to detect, debug, and validate AI outputs, especially in high-risk or ambiguous contexts.
- **Collaborative Integration:** Skill in embedding AI tools into team workflows without undermining human roles or creativity.
- **Ethical Discernment:** Awareness of bias, hallucination risks, and the societal implications of automation.
- **Epistemic Awareness:** Understanding the boundaries of AI reasoning and avoiding overtrust in machine-generated decisions.

These dimensions align with emerging AI literacy frameworks, such as Digital Promise’s triad of *Understanding*, *Evaluating*, and *Using* AI responsibly, and the OECD’s AI literacy domains. Competitional power narrows this lens to the software engineering domain, emphasizing strategic agency in AI-augmented development.

## 5.2 Core Competencies of Hybrid-Literate Graduates

Competency	Description
AI Tool Fluency	Ability to prompt, interpret, and refine AI-generated code
Systematic Debugging	Use AI to assist in testing, but retain manual oversight
Prompt Engineering	Crafting precise queries to elicit useful code or documentation
Epistemic Awareness	Understanding what AI can and cannot reason about
Ethical Judgment	Identifying bias, hallucination, and automation risks
Collaborative Design	Integrating AI into team workflows without undermining human roles

## 5.3 Proposed Curriculum Modules

To operationalize hybrid literacy, universities should introduce new modules and revise existing ones:

### AI-Augmented Software Development

- Hands-on labs using CodeRabbit, Copilot, and other LLM-based tools
- Comparative analysis of human vs. AI-generated code

### Debugging with AI

- Case studies of AI-generated bugs and vulnerabilities
- Techniques for layered testing and validation

### Prompt Engineering for Developers

- Syntax and semantics of effective prompts
- Use of chain-of-thought prompting for complex tasks

### Ethics and Epistemology of AI

- Bias detection in training data
- Limits of AI reasoning in speculative or high-risk domains

### Capstone Projects with AI Integration

- Students build real-world applications using AI tools
- Final reports must include critical reflection on AI's role

## 5.4 Institutional Challenges and Recommendations

Challenges:

- Faculty retraining and resistance to change
- Lack of standardized AI toolkits across institutions
- Risk of students bypassing learning by overusing AI

Recommendations:

- Partner with industry to co-develop AI modules
- Create AI usage policies for coursework and exams
- Encourage reflective assignments on AI’s epistemic limits

## 5.5 Outcome: Restoring Competitional Power

Graduates equipped with hybrid literacy will:

- Adapt to evolving workflows in tech industries
- Collaborate effectively with AI systems
- Retain strategic agency in a market increasingly shaped by automation

This model ensures that AI becomes a tool for empowerment—not displacement.

## 5.6 Comparative Frameworks

Recent efforts to define AI literacy across educational and policy domains offer valuable context for situating competitional power. The Digital Promise framework [16] emphasizes three pillars—understanding, evaluating, and using AI responsibly—highlighting the importance of ethical awareness and critical engagement. The OECD and European Commission’s AILit framework [17] expands this view by outlining 22 competencies across four domains: engaging with, creating with, managing, and designing AI. These models underscore the need for adaptable, reflective learners who can navigate complex AI ecosystems. Stanford University’s AI Literacy Guide [18] adds further nuance by introducing rhetorical and pedagogical dimensions, encouraging learners to interrogate the assumptions embedded in AI outputs. While these frameworks are interdisciplinary in scope, competitional power narrows the lens to software engineering, emphasizing strategic agency, epistemic awareness, and the capacity to integrate AI tools without surrendering human judgment.



## 6 Human Oversight in Critical System Development

As AI tools become embedded in software engineering workflows, their application in critical systems—such as healthcare platforms, financial infrastructures, autonomous vehicles, and defense technologies—raises profound concerns. These domains demand not only technical precision but also ethical accountability, contextual awareness, and resilience under uncertainty. While AI can accelerate development, human oversight remains indispensable to ensure safety, legality, and trustworthiness.

### 6.1 Complexity and Contextual Reasoning

Critical systems often operate under dynamic, high-risk conditions where failure can result in catastrophic outcomes. AI tools, trained on historical data and probabilistic models, struggle to:

- Interpret edge cases or rare scenarios not present in training data
- Adapt to evolving regulatory frameworks or compliance standards
- Understand domain-specific constraints (e.g., medical protocols, legal statutes)

Human developers bring contextual reasoning and domain expertise that AI cannot replicate. [12]

### 6.2 Accountability and Traceability

In regulated industries, traceability of decisions and accountability for outcomes are non-negotiable. AI-generated code lacks:

- Transparent decision-making trails
- Justifications for architectural choices
- Legal liability frameworks

Human intervention ensures that systems are auditable, explainable, and legally defensible. [13] [15]

### 6.3 Ethical and Societal Implications

Critical systems often intersect with human rights, equity, and public trust. AI tools may inadvertently:

- Encode bias in diagnostic algorithms
- Misinterpret culturally sensitive inputs
- Prioritize efficiency over fairness

Human oversight is essential to uphold ethical standards, engage stakeholders, and anticipate societal consequences. [14]

## 6.4 Risk Mitigation and Fail-Safe Design

AI-generated components must be rigorously tested and validated, especially in systems where failure is unacceptable. Human developers are responsible for:

- Designing fallback mechanisms and redundancy layers
- Conducting stress tests and failure simulations
- Ensuring graceful degradation under unexpected conditions

These tasks require foresight, creativity, and judgment beyond the scope of AI reasoning. [11]

## 6.5 Collaborative Intelligence

Recent advancements in AI suggest a shift from automation toward augmentation. Multi-modal AI systems—capable of integrating text, image, audio, and code—are beginning to offer richer contextual understanding and more intuitive interfaces. These developments enhance collaboration by allowing developers to interact with AI in more natural and expressive ways. Additionally, improved prompt engineering techniques, such as chain-of-thought prompting and dynamic prompt tuning, are helping developers elicit more accurate and context-aware outputs. Rather than replacing human developers, these tools are evolving into cognitive collaborators—accelerating routine tasks while deferring to human judgment in complex or ethical scenarios. This trajectory reinforces the value of hybrid literacy: developers who understand both the capabilities and limitations of AI are best positioned to harness its strengths without surrendering control. In this light, human-AI symbiosis is not a compromise—it is a strategic alliance.

Rather than replacing human developers, AI should be integrated as a collaborator—one that augments decision-making but defers to human authority in critical junctures. This hybrid approach fosters:

- Shared responsibility between humans and machines
- Continuous learning loops from human feedback

- Adaptive systems that evolve with human values

In critical system development, the human-in-the-loop model is not optional—it is foundational.

## 6.6 Educational Implications of Critical System Development

The development of critical systems—such as healthcare platforms, financial infrastructures, and autonomous technologies—requires not only technical precision but also deep contextual insight and ethical judgment. While AI tools can assist with documentation, testing, and code generation, the core techniques and decision-making processes involved in building resilient, trustworthy systems still reside in the minds of human developers.

This reality underscores the enduring importance of traditional software engineering and computer science education. Subjects such as systems architecture, fault-tolerant design, cybersecurity, and formal methods provide foundational knowledge that AI cannot replicate or replace. These disciplines cultivate the analytical thinking, domain expertise, and foresight necessary to anticipate edge cases, design fail-safe mechanisms, and ensure compliance with evolving regulations.

Students preparing for roles in critical system development should be encouraged to pursue coursework that emphasizes human-led design, ethical reasoning, and rigorous validation techniques. AI may serve as a supportive tool, but its contributions are limited by its lack of contextual awareness and inability to assume accountability. The future of critical systems depends on developers who combine technical mastery with human judgment—an educational goal that must remain central in university curricula.

# 7 Human-Centric Collaboration in the Age of AI

While AI-assisted tools and automated workflows continue to reshape the software development landscape, the irreplaceable value of human-to-human interaction remains central to successful project execution. In particular, interpersonal communication fosters team cohesion, emotional resonance, and shared ownership—qualities that no algorithm can replicate.

## 7.1 Emotional Connectivity and Team Spirit

Face-to-face or synchronous communication—whether in person or via video calls—helps cultivate trust, empathy, and mutual respect among team members. These interactions build morale and a sense of belonging, which are critical for sustaining motivation and creativity throughout long development cycles.

## 7.2 Enhanced Efficiency in User-Centric Projects

In projects where user involvement is high—such as UX design, agile prototyping, or stakeholder-driven iterations—human dialogue ensures that feedback is not only heard but deeply understood. Nuances in tone, body language, and spontaneous discussion often reveal insights that structured data cannot capture.

## 7.3 Educational Implications for Future Developers

Students pursuing software engineering and computer science should be encouraged to elect subjects that explore human-centered design, communication strategies, and collaborative development methodologies. Courses in psychology, organizational behavior, and user experience design can equip future developers with the interpersonal skills needed to thrive in team-based environments and user-facing projects. These disciplines complement technical expertise and prepare students for real-world challenges where empathy and clarity are just as vital as code.

## 7.4 Complementing AI with Human Insight

Even as AI tools streamline documentation, testing, and code generation, they lack the emotional intelligence and contextual awareness that human conversations provide. Techniques like pair programming, design sprints, and collaborative retrospectives remain vital for aligning goals, resolving conflicts, and adapting to change.

## 7.5 Conclusion

Human-to-human interaction is not a relic of pre-AI development—it is a strategic asset. When combined with intelligent systems, it creates a hybrid environment where efficiency meets empathy, and innovation is grounded in shared understanding.

# 8 Conclusion

The integration of AI into software development is not a zero-sum game. Rather than displacing human developers, intelligent systems are becoming collaborators—accelerating workflows while relying on human judgment for nuance, ethics, and creativity. This paper has argued for the necessity of hybrid literacy: developers must understand both the capabilities of AI and the irreplaceable value of human insight. The addition of human-to-human interaction as a core theme reinforces this perspective. Emotional connectivity, team spirit, and user empathy are not peripheral—they are foundational to successful development. As AI continues to evolve, the most effective teams will be those that embrace

both technological fluency and interpersonal depth. The future belongs to those who can code with machines and connect with humans.

## References

- [1] Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374.
- [2] Pearce, H., Ahmad, M., et al. (2022). Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Suggestions. arXiv:2108.09293.
- [3] Ziegler, J., Liu, J., et al. (2022). Human-AI Pair Programming: Experiences with GitHub Copilot. Microsoft Research.
- [4] Barke, S., Richards, S., & Pradel, M. (2022). Grounded Copilot: How Programmers Use and Assess AI in Code Completion. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2), 1–26.
- [5] Jarrahi, M. H. (2018). Artificial Intelligence and the Future of Work: Human-AI Symbiosis in Organizational Decision Making. *Business Horizons*, 61(4), 577–586.
- [6] Menzies, T., & Williams, L. (2020). Requirements Engineering in the Age of AI: Challenges and Opportunities. *IEEE Software*, 37(4), 7–11.
- [7] Kalliamvakou, E., Bird, C., Zimmermann, T., et al. (2014). The Promises and Perils of Mining GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, 92–101.
- [8] Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. arXiv:2203.02155.
- [9] Selwyn, N. (2019). *Should Robots Replace Teachers? AI and the Future of Education*. Polity Press.
- [10] Shneiderman, B. (2020). Human-Centered AI. *Communications of the ACM*, 63(3), 29–32.
- [11] Karadwa, A. (2023). Enhancing Decision-Making with Explainable AI in Critical Systems. *International Journal of Scientific Research and Engineering Development*, 8(3), 131. [Explores XAI techniques like LIME and SHAP in healthcare, finance, and defense systems]
- [12] Leyli-abadi, M., et al. (2025). A Conceptual Framework for AI-Based Decision Systems in Critical Infrastructures. arXiv:2504.16133. [Proposes a holistic framework for integrating human-AI collaboration in energy, mobility, and aeronautics]
- [13] Frenette, J. (2023). Ensuring Human Oversight in High-Performance AI Systems: A Framework for Control and Accountability. *World Journal of Advanced Research and Reviews*, 20(2), 1507–1516. [Discusses governance strategies to maintain human control over autonomous AI systems]

- [14] Langer, M., Baum, K., & Schlicker, N. (2024). Effective Human Oversight of AI-Based Systems: A Signal Detection Perspective. *Minds and Machines*, 35(1). [Applies signal detection theory to error and fairness detection in AI outputs]
- [15] Ho-Dac, M., & Martinez, B. (2024). Human Oversight of Artificial Intelligence and Technical Standardisation. arXiv:2407.17481. [Analyzes regulatory governance under the EU AI Act and the role of standardization in oversight]
- [16] Mills, K., Ruiz, P., Lee, K., et al. (2024). *AI Literacy: A Framework to Understand, Evaluate, and Use Emerging Technology*. Digital Promise. Retrieved from <https://digitalpromise.org/2024/02/21/revealing-an-ai-literacy-framework-for-learners-and-educators/>
- [17] OECD. (2025). *Empowering Learners for the Age of AI: An AI Literacy Framework for Primary and Secondary Education*. OECD & European Commission. Retrieved from <https://ailiteracyframework.org/>
- [18] Stanford Teaching Commons. (2024). *Understanding AI Literacy: A Generative AI Teaching Guide*. Stanford University. Retrieved from <https://teachingcommons.stanford.edu/teaching-guides/artificial-intelligence-teaching-guide/understanding-ai-literacy>