

Memento Mori: Sustainability of the Software Development Ecosystem in the Age of Agentic AI

José Antonio Parejo Maestre¹ 

SCORELab, Universidad de Sevilla, Spain
japarejo@us.es

Abstract. Programming assistants based on generative models have evolved from syntactic autocompletion to agentic tools capable of reading repositories, editing files, and executing commands (e.g., *Codex CLI*, *Claude Code*). This evolution reshapes the division of labor, to the point that ubiquitous adoption of agentic AI threatens the sustainability of software development as an ecosystem, undermining the training of junior profiles and the acquisition of the competencies needed to build complex systems. Drawing on recent evidence of productivity gains in bounded tasks and on emerging signals of *early-career* job displacement, this *position paper* argues that short-term efficiency must not erode engineering judgment. We rely on Aracil's epistemological distinction between engineering as a capacity for design and decision-making versus mere technique, and we use historical analogies of systemic collapse (Rome, Maya, Qing) to illustrate how local optimization can induce structural fragility. We conclude with a manifesto proposal: a hierarchy of values to guide AI governance and adoption in industry and academia, with the aim of prioritizing human agency and rigorous validation over code generation with immediate results but superficial verification.

Keywords: software engineering · education · generative AI · sustainability · governance · philosophy of engineering

1 Introduction

In 2026, programming assistants have evolved into autonomous agents capable of operating on the command line and the file system, integrating deeply into development environments. This capability displaces tasks historically assigned to junior profiles —unit tests, refactorings, documentation— creating a paradox: we maximize short-term efficiency while degrading the knowledge-transfer mechanism that turns today's juniors into tomorrow's senior engineers.

Research confirms drastic speed improvements for bounded tasks, with time reductions above 50% in certain contexts [12,2]. As a result, a large majority of organizations are currently prioritizing AI integration into their processes [4]. However, recent administrative data in the U.S. already show a decline in hiring for 22–25-year-olds in AI-exposed occupations, while demand for expert profiles remains stable [9].



In parallel, evidence is emerging of adverse side effects of AI integration in software development: degradation in code quality, overconfidence, and, crucially, a negative impact on learning key skills. A controlled experiment by Anthropic recently found that AI-assisted junior developers, while faster, developed 17% less of their code-comprehension and debugging capabilities [14]. Moreover, the study points to a decrease in peer-to-peer technical communication, eroding the transfer of knowledge about software structure and design. Complementarily, the 2025 Stack Overflow Developer Survey suggests that the use of Generative AI fosters patterns of superficial review and “approval fatigue” in the face of plausible outputs [10].

2 Engineering judgment is more than creating valid code

To understand the risk outlined by these studies, we must define what is lost when AI is used for every implementation in a system without proper human direction and supervision (even when the generated implementation is functionally correct). Following Javier Aracil, engineering is not merely the application of science or technique (the “doing”), but an activity of will and design aimed at transforming reality to satisfy human needs under conflicting constraints [1].

Agentic AI is, in Aracil’s terms, a formidable technical tool for the *fabrication* of code, but, at least in its current state, it lacks the capacity for *design*. We understand design as the synthesis of non-functional, economic, and ethical trade-offs. If the engineer loses the ability to understand the characteristics and properties of the “material” (code) by delegating its manipulation and creation to an agent, they also lose the ability to exercise the critical judgment required to validate whether the system fulfills not only its purpose but also the balance among those trade-offs. Ecosystem sustainability depends on keeping humans in the decision loop with knowledge of cause, capable of a critical and holistic evaluation of the implementation.

3 Historical Analogies: Lessons on Systemic Fragility

The history of complex systems shows that collapse is rarely sudden or mono-causal. But it is often the result of optimizations that erode long-term resilience. We propose three analogies to understand the risk of embracing the paradigm of “automatic development” over “governed development with human validation”:

Cost optimization can sometimes lead to collapse (late Rome): In the Empire’s final centuries, Rome increased its dependence on allied contingents and non-Roman recruits (often grouped under the label *foederati*), integrating them into campaigns and, in some cases, treating them as stable sources of recruitment [8]. *Parallel:* If we outsource all the “dirty work” of code to AI agents (our digital *foederati*) without rigorous validation of their outputs—and during the early stages of engineering careers—we risk producing engineers unable to “fight” (debug and understand behavior in depth) when the system fails critically, or when AI is unable to solve a problem by proposing a valid solution.



The high-level equilibrium trap (late imperial China): Mark Elvin formulated the “high-level equilibrium trap” to explain how, in certain contexts of preindustrial China, the efficiency of productive methods and the relatively low cost of labor reduced economic incentives to invest in capital substitution and transformative innovations [6,5]. *Parallel:* AI allows us to refine today’s software production at low cost, but, by its nature (learning from a corpus built in the past), it tends to reinforce existing technologies and solutions. In a world where implementation is delegated and human review is shallow, what incentive remains to propose new paradigms or design novel programming languages?

The Maya collapse (multicausality and socio-ecological stress): The literature on the collapse of the Maya empire emphasizes that it was not a single event, but a multicausal process in which ecological pressures (e.g., droughts), political dynamics, and socioeconomic fragilities interacted [3,7]. *Parallel:* A software development ecosystem with an “elite” of seniors who only orchestrate agents, without a solid base of juniors who understand the “agriculture” of code (maintenance, debugging, deep validation) and who mature through learning in a team with setions, is exposed to a generational collapse.

Memento mori: In the triumphal parades of ancient Rome, a servant whispered to the victorious general: “*Memento mori*” (remember that you are mortal). The mission of this article is to play that role for the software engineering community: to remind us that our technical capabilities —and engineers—are mortal, and difficult to develop and educate. We must prioritize the long-term viability of the profession and the maintenance of the development ecosystem over dazzling gains in immediate productivity that push us to introduce AI in the early stages of education and to replace juniors in training with seniors commanding hordes of agents.

This is not a corporatist or nostalgic plea to preserve a profession for its own sake. Software engineering is not an end in itself; it is a critical means to keep technique governed by human will. Following Aracil’s distinction, engineering is “will to design”: the capacity to make and justify decisions under constraints and trade-offs. If we let agentic AI absorb the junior learning pathway, the risk is not merely the disappearance of a trade, but the transmutation of engineering into a class of technological scribes. Medieval scribes could copy sacred texts flawlessly without understanding a word. Likewise, we risk producing a generation of “engineers” who can only shallowly supervise functionality (technique) while losing the ability to comprehend and validate substance (design and trade-offs). Protecting the ecosystem—the profession and its cognitively frictionful learning pipeline—is therefore the only robust guarantee that the software that runs the world does not become an ungovernable black box, fragile in the face of failure.

4 A Manifesto for the Age of Agentic AI

Inspired by the structure of the Agile Manifesto, we propose a hierarchy of values to address the challenges illustrated above. We recognize the value of the



We value	over
1 Human agency and accountability	Opaque efficiency without traceability
2 Explicit design decisions	Functionally correct auto-generated implementation
3 Critical evaluation and deep validation	Practical/functional plausibility
4 Evidence-based rigorous validation	Delivery speed
5 Learning through cognitive friction	Delegation convenience
6 Team social capital (mentoring & collaboration)	Brute-force computational power
7 Structural prudence (Zero Trust)	Tool/agent autonomy

Table 1. A hierarchy of values for the use of agentic AI by software engineers, available for signature at <https://github.com/japarejo/sustainable-software-engineering-with-agentic-ai-manifesto> [11]

items on the right, but we value those on the left more in order to ensure the sustainability of software engineering ecosystem.

We do not dismiss the improvements and possibilities offered by agentic AI; on the contrary, we believe software engineers should adopt them responsibly, and that aspiring engineers should learn to use them as part of their toolbox. However, we argue that their application —specially in the early stages of engineering education— must be subordinated to the development and reinforcement of the principles and capabilities described in Section 2, so that automation amplifies engineering judgment rather than replacing it.

Finally, we argue that agentic AI and engineering attitudes, disciplines, and capabilities are not antagonistic, but complementary and potentially synergistic; precisely for that reason, we must work to ensure they reinforce each other. Recent approaches such as *spec-driven development* suggest a promising path in which automation relies on specifications, intent, and explicit validation criteria, elevating the engineer’s role from writing code to design and decision-making within a framework of rigorous evaluation and validation [13].

5 Conclusion

Software engineering faces its own “Memento Mori” moment: as software has become the new oil that makes the world move, it must remember its mortality and fragility in the face of the seduction of absolute automation and drastic optimization of development costs. Ecosystem sustainability requires prioritizing the formation of judgment —design, evaluation, and validation— over the amplification of capacity, especially in the early stages of training and professional careers. If we allow AI to write software without cultivating the human capacity to judge it, we will cease to be engineers and will end up as mere spectators, jaded critics of its functioning, wholly opaque to our minds, and defenseless in the face of the failure of a system we do not understand and can no longer debug.



Acknowledgments. This work is part of SCORELab's research activities.

Disclosure of Interests. The author declares no conflicts of interest beyond a genuine concern for the future of the profession.

References

1. Aracil, J.: La ingeniería como actividad humana: reflexiones sobre el diseño y la decisión. Real Academia de Ingeniería (2010)
2. Cui, K.Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., Salz, T.: The productivity effects of generative ai: Evidence from a field experiment with github copilot. MIT Generative AI Research (working paper / release) (2024), <https://mit-genai.pubpub.org/pub/v5iixksv>
3. Diamond, J.: Collapse: How Societies Choose to Fail or Succeed. Viking (2005)
4. DORA: Dora report 2024. Online report (2024), <https://dora.dev/research/2024/dora-report/>
5. Elvin, M.: The high-level equilibrium trap. Economic Organization in Chinese Society. Stanford (1972)
6. Elvin, M.: The Pattern of the Chinese Past. Stanford University Press (1973)
7. Hoggarth, J.A., et al.: Drought and the maya civilization (evidence and societal impacts). Quaternary Science Reviews (2017), peer-reviewed synthesis connecting drought evidence with Maya-region societal change.
8. Jones, A.H.M.: The later Roman Empire, 284-602: a social economic and administrative survey, vol. 1. JHU Press (1986)
9. Lab, S.D.E.: Canaries in the coal mine? six facts about the recent employment effects of artificial intelligence (2025), https://digitaleconomy.stanford.edu/app/uploads/2025/11/CanariesintheCoalMine_Nov25.pdf
10. Overflow, S.: Stack overflow developer survey 2025 — ai (2025), <https://survey.stackoverflow.co/2025/ai>
11. Parejo Maestre, J.A.: Sustainable software engineering with agentic ai manifesto, v1.0. Zenodo / GitHub repository release (Feb 2026). <https://doi.org/10.5281/zenodo.18750578>, <https://zenodo.org/records/18750579>, version 1.0 released 23 February 2026; available at <https://github.com/japarejo/sustainable-software-engineering-with-agentic-ai-manifesto/tree/v1.0>
12. Peng, S., Kalliamvakou, E., Cihon, P., Demirer, M.: The impact of ai on developer productivity: Evidence from github copilot. arXiv preprint arXiv:2302.06590 (2023), <https://arxiv.org/abs/2302.06590>
13. Piskala, D.B.: Spec-driven development:from code to contract in the age of ai coding assistants (2026), <https://arxiv.org/abs/2602.00180>
14. Shen, J.H., Tamkin, A.: How ai impacts skill formation (2026), <https://arxiv.org/abs/2601.20245>

