MIT Sloan Management Review

The Hidden Costs of Coding With Generative AI

Edward Anderson, Geoffrey Parker, and Burcu Tan

Generative AI can boost coding productivity, but careless deployment creates technical debt that cripples scalability and destabilizes systems.

August 18, 2025
Reprint 67110

sloanreview.mit.edu

The Hidden Costs of Coding With Generative AI

Generative AI can boost coding productivity, but careless deployment creates technical debt that cripples scalability and destabilizes systems.

By Edward Anderson, Geoffrey Parker, and Burcu Tan

GENERATIVE AI CAN BE A POWERFUL productivity booster in coding — but only when deployed thoughtfully. Used carelessly, it can cripple scalability, destabilize systems, and leave companies worse off.

Generative AI is growing explosively across knowledge work, particularly in software development. OpenAI's latest release, GPT-4.1, focuses heavily on enhancing coding capabilities and is a step toward full automation. Organizations adopting these tools are anticipating major gains. And early research supports their optimism: GitHub has reported that programmers using Copilot are up to 55% more productive, and McKinsey has found that developers can complete tasks up to twice as fast with generative AI assistance.

But these positive indicators come with a major caveat. The studies were conducted in controlled environments where programmers completed isolated tasks — not in real-world settings, where software must be built atop complex existing systems. When the use of AI-generated code is scaled rapidly or applied to brownfield (legacy) environments, the risks are much greater and much harder to manage. As part of our ongoing research on the strategic management of AI-augmented software development, we conducted interviews with individuals involved in developing software — ranging from junior developers to lead software engineers and CIOs — across a diverse set of industries, including insurance, web hosting, social media, defense, management consulting, and fintech. Drawing on insights from these interviews, a review of the trade press, and our own economic modeling, we have identified several strategic trade-offs that companies should consider when adopting generative AI for software development.

Why Technical Debt Grows Faster With AI

When an organization rapidly introduces new software into existing systems, it can inadvertently create a tangle of dependencies that compounds its *technical debt* — that is, the cost of additional technological work that will be needed in the future to address shortcuts taken and quick fixes made during development. Technical debt is the hidden underbelly of digital technology. It is the 60-year-old COBOL code in banking systems that was never properly documented or updated. It is the shortcut of representing the current year with two digits instead of four, leading to the Y2K crisis, which cost hundreds of billions of dollars to fix globally. The buildup of technical debt causes slower development cycles, increased complexity, and security vulnerabilities, potentially leading to system failures.

The Consortium for Information & Software Quality estimates the cost of technical debt in the U.S. to be at least $2.4 trillion. Despite this exorbitant price tag, most organizations do not prioritize dealing with technical debt, with the majority allocating less than 20% of their tech budget to paying it down. Developers we interviewed admitted that they often "sneak in" technical debt management during maintenance because leadership rarely approves dedicated time for it. As one senior developer put it, "No one fixes the technical debt, which then causes more fires, which prevents you from fixing the technical debt, and so on."

You can think of technical debt as operating much like financial debt. The "principal" is the work needed to modernize and refactor code; the "interest" is the ongoing complexity tax that slows maintenance, complicates scaling, and raises the risk of failure. While some debt is unavoidable, implementing AI-generated code is often akin to borrowing at a much higher interest rate. As one of the developers we interviewed said, "The problem with AI is that it can't see the big picture." Developers we interviewed also told us about code duplications,

Legacy systems already tend to carry hidden debt; layering AI-generated code on top of them creates additional tangled dependencies.

integration problems, dependency conflicts, a lack of context awareness, and myriad other problems that come with coding with AI. Indeed, when GitClear analyzed millions of lines of code from 2020 to 2024, it uncovered an eightfold increase in duplicated code blocks and a twofold increase in code churn — both measures of declining code quality. The "2024 Accelerate State of DevOps" report from Google's DevOps Research and Assessment team found that a 25% increase in AI usage improves code review and documentation but results in a 7.2% decrease in delivery stability. So, what looks like rapid progress today could turn into costly setbacks tomorrow.

Adding AI-generated code into brownfield environments magnifies these risks. Legacy systems already tend to carry hidden debt; layering AI-generated code on top of them creates additional tangled dependencies that slow future development and destabilize systems even more. As one engineer at a top three AI company told us, "AI can't see what your code base is like, so it can't adhere to the way things have been done." Future AI models may be able to analyze entire code bases and help solve these problems, but for now, working in brownfield environments makes it much more likely that AI-generated code will compound technical debt.

Letting technical debt compound is dangerous. Southwest Airlines' 2022 meltdown — which stranded over 16,900 flights and cost the airline over $750 million — was rooted in technical debt in its crew-scheduling systems. Technical debt drove the massive 2024 CrowdStrike outage that led to worldwide failures in health care delivery. In May 2025, Newark Liberty International Airport in New Jersey was plagued by massive delays and hundreds of flight cancellations that were caused by a combination of antiquated air traffic control technology and staffing shortages. Failures like these show how invisible risks can suddenly cripple even major organizations. Without deliberate efforts to "pay down the principal," organizations risk becoming overwhelmed — first slowly, then all at once.

When It's (Relatively) Safe to Use Generative AI for Coding

The potential risks don't mean that companies should always avoid using generative AI for coding. In the right contexts, such as rapidly prototyping new products in a greenfield (new) environment, AI-generated code can deliver a real speed advantage. In these cases, early-stage code will likely require major revisions anyway, making technical debt less costly.

But when scalability is a priority, or within brownfield environments weighed down by legacy systems, AI-generated code must be deployed with extreme care. Two factors strongly influence the level of risk:

- **The development environment (greenfield versus brownfield):** Greenfield projects, with no legacy code, involve lower risk. Brownfield projects are far more vulnerable to hidden debt accumulation.
- **Software engineering skills:** Our interviews and accounts by senior developers suggest that low-skilled software developers are more likely to let AI-generated technical debt snowball. Highly skilled developers are better equipped to recognize architectural flaws and mitigate technical debt before it spreads. As a software developer in a Fortune 50 tech company's AI infrastructure area shared, "[With AI] a junior engineer can write as fast as a senior engineer, but they don't have the cognitive sense of what they're doing … or what problems they're causing … or even if it's a good idea to do what they're doing."

Managers should exercise caution when inexperienced developers are deploying AI-generated code, or when such code is being deployed in a brownfield environment. When both risk factors are present, it may be best to avoid deploying AI-generated code entirely.

Reducing the AI 'Tax' on Technical Debt

Even as generative AI continues to improve in its usefulness, our research indicates that the associated risks will remain important.

Organizations must treat AI tools' tendency to increase technical debt as a strategic risk, not just an operational nuisance.

Organizations must treat AI tools' tendency to increase technical debt as a strategic risk, not just an operational nuisance. To fully realize generative AI's promise, companies must do the following:

- **Develop clear guidelines for when and how to use AI-assisted coding tools.** Many large companies (including Microsoft, Google, Meta, and Salesforce) have already established responsible AI use policies grounded in ethical principles such as fairness, privacy, and inclusiveness. However, translating these high-level ideals into actionable, day-to-day guidelines for AI-augmented software development is an ongoing process. We expect that these guidelines will soon expand to include using AI to improve existing code bases. As AI-assisted coding tools evolve, tackling technical debt is likely to become an important use case for the technology. One of the software developers we interviewed was optimistic that "if you capture the logic and train AI, you can reduce technical debt rapidly." There is already some empirical evidence of AI's potential use in maintaining legacy code, but clearly defining tasks and keeping a human in the loop will be key. Notably, Morgan Stanley has been experimenting with this approach using an in-house GenAI tool because off-the-shelf models are not yet capable of handling legacy code translations effectively.
- **Treat technical debt management as an engineering priority, not an afterthought.** There are many guidelines for technical debt management. What matters most is building it into everyday workflows rather than just scrambling to fix something when it breaks. Otherwise, according to our economic modeling, performance will see a brief short-term increase, but technical debt will eclipse this improvement in the long term.
- **Invest in training junior developers so that they are better able to use AI tools without creating excessive technical debt.** Several companies have already begun upskilling efforts, particularly in prompt engineering, through in-house training or external workshops. But developing the ability to assess AI-generated output requires a different approach. This is where mentorship becomes essential. Traditional code reviews must evolve. Senior developers should not only evaluate code quality but also coach junior team members in responsible and effective AI use. This kind of guidance can also serve as a guardrail against the erosion of next-generation developers' foundational skills.

Generative AI is here to stay. But like any powerful tool, it demands respect, discipline, and strategy. Organizations that rush ahead blindly risk finding that today's productivity gains come at the cost of tomorrow's ability to compete.

Edward Anderson is the Betty and Glenn Mortimer Centennial Professor for Business at the University of Texas McCombs School of Business and the University of Texas Supply Chain Management Center. Geoffrey Parker is the Charles E. Hutchinson '68A Professor of Engineering Innovation at Dartmouth College and faculty director for the Arthur L. Irving Institute for Energy and Society. He is also a research fellow at the MIT Initiative on the Digital Economy. Burcu Tan is an associate professor at the University of New Mexico Anderson School of Management.

https://doi.org/10.63383/hadW7619.