How organisations are reckoning with AI addiction

The Hind, dt: 07.06.2026, pg.no. 23

DECODED

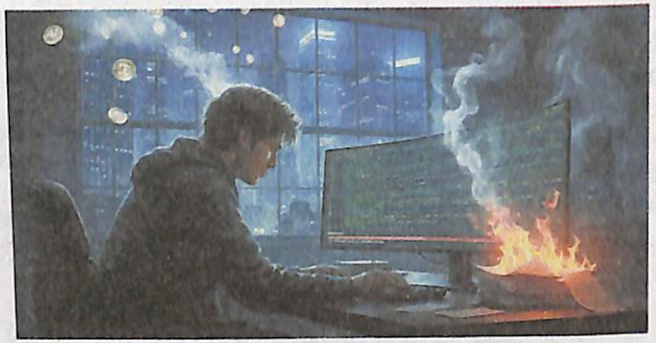
John Xavier

In December 2025, Microsoft opened the doors for thousands of its engineers to use Anthropic's Claude Code – an AI coding assistant – as part of an internal experiment to benchmark the best tools available. That test worked too well, as some employees decisively chose it over Microsoft's own AI coding assistance – GitHub Copilot CLI.

Then, by May 2026, the software giant began cancelling most of those Claude Code licences, instructing teams behind Windows, Microsoft 365, Outlook, Teams, and Surface to migrate back to Copilot CLI before June 30. That day is apparently the last day of the company's fiscal year. While it has called this rollback a "toolchain unification," the financial reality, per multiple reports, suggests that engineers had used Claude Code a little too much.

This is the paradox of enterprise AI adoption in 2026. To understand this situation better, it helps to go down the memory lane on how software companies operated earlier. Before OpenAI's ChatGPT arrived in late November 2022, the dominant tools in a developer's toolkit were a code editor like VS Code, a version control system like Git, documentation wikis, Jira boards, Slack threads, and an occasional Stack Overflow lookup.

Most of these were open-source tools or flat-rate subscriptions that cost essentially nothing. An engineer could work all day without the company spending an extra rupee.



Every action an agentic tool does consumes tokens — the unit of text a language model processes. AI-GENERATED IMAGE

Then came the chatbots like ChatGPT, Claude, Gemini, Llama, and a cascade of capable successors. GitHub Copilot, which had been available since 2022, gained significant traction as models improved. By 2024, engineering teams were routinely asking AI assistants to write boilerplate, review pull requests, suggest fixes, and generate test cases. As these interactions were short and mostly conversational, the costs were modest.

But the arrival of agentic coding tools disrupted the way engineers worked as they didn't just answer queries but took a series of actions. These AI agents could read files, run tests, check for errors, revise code, read the context again, and loop through all of this autonomously. Claude Code, Cursor, and their peers work this way.

The problem comes down to what is called "token maxxing." Every action an agentic tool does consumes tokens – the unit of text a language model processes.

A session that starts at five thousand tokens per call can balloon to two hundred thousand tokens per call by the time the agent is fifty turns deep. As one detailed analysis documented, a developer who tracked token consumption across 42 agentic runs found that roughly 70% of tokens consumed were effectively wasted. The agent was re-reading files it had already processed, exploring dead ends, and re-verifying things it already knew. The corporate toll of this wastage is becoming public in uncomfortable ways.

An internal leaderboard that ranked teams by AI usage volume had accelerated adoption, which accelerated bills. Uber's COO, Andrew Macdonald admitted that drawing a line between rising token consumption and actual consumer feature delivery was proving elusive.

Box CEO Aaron Levie has argued that the token consumption problem will eventually spread from engineering into legal, sales, and every other knowledge-work function.

Goldman Sachs, in a report, projected that global token consumption could increase 24-fold by 2030 as agentic AI becomes standard practice.

It is unclear when these perspectives will converge. But, for now, it is clear that companies are becoming thrifty when it comes to token usage.

Agentic AI tools are becoming an infrastructure cost that scales with every autonomous decision an agent makes. And organisations that do not implement usage controls, token budgets, and honest ROI benchmarks are not adopting AI, but are funding it.