ChatGPT

The Resurgence of the Hands-On Technical Manager

By Dinis Cruz & ChatGPT

The Faustian Bargain: Trading Code for Management

For decades, many organizations inadvertently pushed their best technical people into management roles – often the only path for career advancement [1]. This promotion came with a Faustian bargain: in exchange for a leadership title, the individual would step away from hands-on technical work. Talented engineers who once thrived on coding and architecture found themselves drowning in meetings and spreadsheets, fearing they'd **"lose their technical edge"** once they donned the manager's hat [2]. In the fast-paced tech world, this fear is justified – technology changes so quickly that stepping away even briefly can leave a former guru feeling obsolete [2].

Career Paradox: It often seemed like the reward for being a great engineer was **not** getting to do engineering anymore. Organizations would take an expert developer or sysadmin and say, "Congratulations, you're in charge now – please stop coding and start managing people." While some professionals enjoy this switch, many others make a *Faustian deal* they later regret. They surrender the very technical skills that made them valuable, risking a disconnect from the craft they love. Over time, such managers can become what engineers jokingly call "post-technical" – leaders who no longer truly understand the work being done on the ground. It's an ironic loss for the company as well: the *institutional knowledge* and deep expertise of these individuals fade just when they ascend to positions where that insight could have guided strategy.

To be fair, not all companies have clung to this one-track model. Some forward-thinking organizations introduced dual career ladders, allowing a "technical track" parallel to the management track. This means an outstanding engineer can become a **Principal Engineer, Architect, or Technical Fellow** – high-status roles without direct people management [3]. These frameworks ensure that talented technologists **"can reach highly influential roles without having to switch into leadership"** [3], thereby retaining their hands-on focus. Yet, even in companies with such options, the allure of leadership or the promise of greater compensation often funnels people into management before they're ready – and they may pay for it by losing touch with the technology.

One of the co-authors of this article experienced this dilemma firsthand. Rather than taking the management fast-track in his 20s or 30s, he deliberately **stayed technical** for decades, only moving into a CISO (Chief Information Security Officer) role in his 40s – by which time he was a seasoned engineer. This was unusual; many peers had long since gone "up the ladder" and stopped coding. But in his case, *the payoff was clear*: upon becoming an executive at a large company, he could still script and automate solutions, understand cloud architectures, and dive into technical discussions with ease. Instead of feeling out of depth, he brought an **engineering mindset** to leadership. His journey is not unique – a number of leaders in the industry who followed a similar path found that retaining real technical skills made them far more effective in management. It highlights a crucial point: a leader with authentic, up-to-date technical insight can steer teams with a clarity and efficiency that pure-play managers often struggle to match.

Why Technical Skills Still Matter in Leadership

Technical expertise isn't just an individual contributor's game – it's a force multiplier in leadership. Here are some key reasons why keeping (or developing) technical skills makes for better managers and executives:

- **Realistic Planning & Estimation:** Managers who have built and shipped systems before can **gauge the true complexity** of tasks. They know from experience which problems are straightforward and which are likely to explode into month-long nightmares. This leads to more accurate timelines, sensible budgets, and fewer nasty surprises. In fact, an ex-engineer-turned-manager uses their background to judge what's feasible or not, rather than just accepting whatever they're told [4]. That means when an engineer says a feature will take two weeks, a hands-on leader can tell if it's a two-day script or a two-month research problem.
- **Credibility & Trust:** Technical teams tend to respect leaders who *speak their language*. A manager who understands the nuances of database scaling or can read code earns instant credibility. Developers don't have to oversimplify (or hide) issues, and they're less likely to view the boss as out-of-touch. Conversely, one common fear of new engineering managers is **"being deemed 'post-technical'"** [4] – essentially losing credibility with the team. Retaining a technical edge helps avoid that. When your team knows you could review a pull request or debug a tricky outage if needed, there's a baseline of trust that boosts morale and communication.
- **Efficient Problem-Solving:** Rather than waiting for reports to bubble up through layers, technically adept leaders can jump in to *diagnose problems or explore solutions directly* when time is of the essence. This doesn't mean they micromanage or write all the code themselves, but it means they can pair with an engineer on a critical issue and add value. They can read an error log, understand a design diagram, and even prototype a quick script if necessary. This hands-on help can save days of back-and-forth. It also guards against being misled – a savvy tech manager can't have the wool pulled over their eyes by jargon, because they **truly understand the work**.
- **Strategic Insight into Technology:** A leader who stays close to technology is better positioned to recognize important shifts – and capitalize on them. Whether it's cloud computing, containers, or machine learning, they can separate hype from reality. For example, when cloud services first emerged, many traditional managers saw them as "magic" outside their comprehension. But those with deeper technical grounding could assess cloud offerings and guide their companies to adopt them wisely (or avoid pitfalls). The same goes for today's AI and blockchain waves. Technical leaders can align **technical decisions with business strategy** more effectively, because they grasp both the high-level vision *and* the implementation details.

Wardley Mapping Mindset: In addition, a technically grounded leader can modulate their management style based on the lifecycle of the product or project. We can borrow Simon Wardley's framework of *Pioneers, Settlers, and Town Planners* (also termed *Explorers, Villagers, and Town Planners) to illustrate this. In early, uncertain stages, teams need to behave like explorers – experimenting freely, accepting failures, and discovering what works. Later, as ideas mature, teams switch to a villager mentality – turning prototypes into reliable products. Finally, in well-understood domains, town planners** establish structure, efficiency, and scale. An experienced technical manager instinctively knows when the team should be in exploration mode versus execution mode. Because they've lived through the full software development lifecycle, they encourage innovation when it's time to innovate, and enforce rigor when it's time to standardize and scale. They can pivot from brainstorming one week to instituting strict QA processes the next, reading the needs of the project. This adaptability is hard to achieve for someone who hasn't "been there, done that" on the technical side. It's one more way technical insight makes for more effective leadership [5].

Figure: Illustration of the Explorer, Villager, and Town Planner archetypes in an organization's workflow. Effective technical leaders recognize when their teams should behave like explorers (pioneering new ideas in uncharted territory) versus villagers (iterating and building out a proven concept) or town planners (optimizing and industrializing at scale). This concept, derived from Simon Wardley's mapping framework, highlights the value of aligning management style to the stage of innovation [5] *.*

The Cost of Going "Post-Technical"

What happens when a leader loses their technical edge? We've all seen it: a manager or executive who used to be an ace coder a decade ago but hasn't written a line since Java 1.4. The consequences of going "post-technical" can range from frustrating to catastrophic:

- **Out-of-Touch Decision Making:** Leaders with outdated knowledge may treat modern technology as a black box – or worse, fall for buzzwords without understanding the substance. They might label new developments like serverless computing or neural networks as "magic" that the kids are doing these days. This lack of understanding can lead to **poor decisions**: over-investing in a fad, or dismissing a game-changing innovation as impossible just because they don't grasp it. We saw this with cloud adoption – some managers who hadn't kept up thought moving to AWS was trivial (or conversely, thought it was impossible to trust), whereas those still in touch with tech knew how to weigh the pros and cons. In cybersecurity, leaders who didn't stay current with modern threats and tools often underestimated risks or failed to implement critical controls, simply because they didn't understand the technical specifics.
- **Unrealistic Expectations:** A manager who hasn't built software in years might seriously underestimate how hard it is to implement a feature or fix a legacy system. From afar, everything looks easier than it is. This can result in setting **impossible deadlines** and overloading teams. For example, if a boss stopped hands-on work back in the 2000s and never dealt with, say, a complex microservices architecture, they might not realize why adding a simple-sounding feature now requires touching five different services and coordinating a dozen API contracts. We've heard of executives expecting a major data integration to be done "in a few days" when any current engineer knows it's a multi-month project. Such disconnects sow frustration and burnout in engineering teams.
- **Erosion of Team Confidence:** Technical staff quickly learn when a boss "gets it" and when they don't. If the team constantly has to translate basic concepts or feels their leader doesn't appreciate the complexities they face, respect erodes. The term "pointy-haired boss" from Dilbert cartoons – a clueless manager meddling in tech he doesn't understand – becomes the unspoken

label. Talented engineers may leave teams led by such managers, seeking environments where leadership has real insight. On the flip side, when managers stay technically literate, teams feel supported rather than second-guessed. It creates a culture where engineers and management can collaborate as equals in problem-solving, instead of talking past each other.

- **Slower Response to Change:** Because a non-technical (or no-longer-technical) manager relies entirely on others for technical input, they add latency to decisions. They must call meetings, consult experts, and defer to committees for every new issue. In contrast, a tech-savvy leader can often cut through the noise. They can personally read up on a new tool, quickly prototype a solution, or grasp an engineer's proposal without needing a week of explanations. In a crisis – say a critical outage or a security incident – time is of the essence. Leaders who remain technically sharp can **make quick, informed decisions** under pressure, whereas those who aren't may be paralyzed without their experts on hand.

There's also a personal cost to consider. Many people go into technology because they *enjoy* building and learning. When they ascend to management under the old model, they may secretly miss the hands-on work. They remember the satisfaction of solving a tough bug at 2 AM or deploying a feature that users love. A decade into pure management, those experiences are memories. Some ex-engineers lament becoming what one might call "PowerPoint executives" – spending all day on slides and budgets, detached from the technical excitement that once motivated them. This isn't to say management isn't rewarding – it's just *different*. And for those who are truly passionate about technology, being forced away from the craft can lead to regret or stagnation. It's high time we find ways to let leaders keep a **foot in both worlds** – to manage effectively *and* stay connected to the tech. As we'll discuss next, that is finally becoming possible.

How AI and Automation Empower Technical Leaders

Until recently, one reason managers had to step back from coding was sheer bandwidth. A director overseeing multiple teams simply doesn't have the hours in the day to read every code change or write software on the side. But the rise of **AI-assisted development** and automation is changing the game. We now have tools that can do in minutes what used to take engineers days – meaning a manager can stay in the loop without imposing huge overhead on the team. This is paving the way for a resurgence of the hands-on technical manager. Here's how modern technology is enabling leaders to retain (or regain) their technical edge:

- **Rapid Code Summarization:** Large Language Models (LLMs) like GPT-4 can digest large codebases or complex technical documents and produce **plain-language summaries**. For a manager who doesn't have time to review every commit, this is a godsend. They can ask for a high-level explanation of a module or an API in terms they understand. For instance, if the team is coding in Rust but the manager is more familiar with Python, an AI assistant could translate the Rust code's intent into a Python-like pseudocode or just into English descriptions. This isn't science fiction – *AI tools now "convert complicated thought processes into simple, common language," bridging the gap between code and dialogue* [6]. A product owner or a non-coding architect can finally get meaningful insight into the code without pestering the developers for a walkthrough.
- **Automated Documentation and Diagrams:** Creating documentation has always been a chore that takes valuable time. It's often the first thing to be neglected when a team is busy. In the past, if a VP asked for an architecture overview or some technical briefing, the team might groan at the days of work needed to prepare slide decks and docs for a non-coding audience. Today, AI-driven documentation tools can **generate accurate documentation in seconds** [7]. Need a quick UML diagram of your microservice architecture? An AI tool can analyze the code or deployment config and spit one out. Need Javadoc-style comments for all your methods? AI can

draft them automatically. *Documentation that would take hours of a developer's time to write can be produced almost instantly by AI* [7] *.* This means a technical manager can **ask for an up-to-date briefing** on the system and actually get it without guilt – the team can leverage automation instead of burning cycles writing docs from scratch. The net effect is better-informed leaders *and* developers who are free to keep coding rather than explaining.

- **On-Demand Prototypes and Analysis:** Managers often have ideas or what-ifs that they hesitate to burden the team with. ("Could we maybe automate X?" "What if we tried Y approach – would it be faster?") In the past, these questions either remained unanswered or required pulling an engineer off-task to investigate. Now, thanks to AI coding assistants, a manager can do a bit of exploration solo. As one technical VP described, *"AI coding tools are perfect for [a manager's] chaotic existence" – you can prompt an AI to whip up a quick prototype in minutes, even between back-to-back meetings* [8] *.* For example, a CTO might use a tool like GitHub Copilot or Cursor to generate a sample script that tests a new API, or to parse some data, without involving the team at all. These throwaway prototypes can validate an idea (or show that it won't work) much faster. It's a way for leaders to **stay hands-on in micro-doses**. Even with just 15 minutes, an AI helper can turn a rough idea into a running code snippet [8] . This keeps the manager's coding muscles flexed and often produces useful insights that benefit the team.
- **Language and Legacy Translation:** In tech, people often specialize in certain programming languages or systems over their career. A manager might be a guru in C++ or COBOL, but the team now writes everything in Go or Kotlin. This can create a language barrier that discourages the leader from delving into code. Here, AI can act as a translator. Modern LLMs can take code in one language and explain its logic in another language (or in plain English). For instance, if a legacy system is in COBOL, an AI could help translate parts of it into Python pseudocode so a new-generation manager can understand it. Conversely, if a system uses cutting-edge tech that the leader isn't yet familiar with, they can **ask the AI to explain "like I'm 5"** and get a gentle intro. This *dramatically lowers the friction* for a manager to engage with unfamiliar technical materials. It's like having a patient tutor on call 24/7. As a result, those who were afraid they'd "age out" of the industry can much more easily **keep up with new paradigms**, guided by AI assistance.

Crucially, all these AI-powered workflows don't replace the team's work – they **augment it**. The goal isn't for the manager to bypass the team or do everything themselves, but rather to keep their understanding sharp and provide informed guidance. It also enables a new kind of request a leader can make to the team: instead of asking engineers to do mind-numbing documentation or status reports, a manager can ask them to integrate AI into their processes. For example, a forward-looking manager might encourage the team to set up continuous documentation generation (so that every time code is committed, the docs update automatically). Then the manager can review those docs to follow progress. Or a manager might ask for a "manager's dashboard" – some lightweight interface (even just a collection of scripts or API endpoints) that they can use to **test and validate** what the team is building. In one real scenario, a tech-leading CISO built small abstraction layers on top of his teams' work so he could write custom queries and verify security controls on his own. Now, with LLMs, a lot of that scaffolding can be built much faster or even on-the-fly.

The big picture is that the **overhead of staying technical has decreased**. What used to require, say, pulling an engineer away for two days to explain a system, can now be done in a 30-minute AI-augmented review session. This means a leader can keep much closer to the tech pulse of their organization without hindering productivity. It's a virtuous cycle: the team benefits from having an engaged, knowledgeable leader, and the leader satisfies their curiosity and passion for technology *while still focusing on their managerial duties*. AI and automation are essentially offering a second chance at the technical path for those in leadership.

Embracing the New Dual-Track Leader

The convergence of these trends – recognition of dual career tracks, and AI tools that blur the line between management and engineering – has set the stage for a new kind of leader. The **resurgence of the hands-on technical manager** isn't about returning to the days of micromanaging or refusing to delegate; it's about **blending deep technical insight with skilled people leadership**. Organizations stand to gain a lot by encouraging this blend:

- **Better Decisions, Faster:** When the people making strategic decisions truly understand the technology, their decisions tend to be more sound. They can foresee the impact of choosing one tech stack over another, or realistically appraise the risks of a project. And when urgent issues arise, they can cut through bureaucracy and make calls quickly because they don't need everything translated for them. In an era where every company is to some degree a tech company, having tech-savvy leadership is a competitive advantage.
- **Stronger Team Alignment:** Imagine a workplace where engineers and their managers are aligned in purpose and understanding. In such an environment, goals are clearer and communication flows freely. There's less "us vs. them" between the devs and management. When leaders roll up their sleeves (figuratively or literally) and partake in technical brainstorming or code review, it reinforces that *everyone is on the same team*. It can also inspire the team – knowing that their manager understands their work often motivates engineers to rise to the occasion and do their best. Mentorship goes both ways too: junior developers can learn from a senior leader's experience, and the leader stays fresh through interaction with the team's ideas.
- **Retaining Talent and Expertise:** By providing a path for technical folks to move up without losing their passion, companies can retain senior talent who might otherwise leave. In the past, a brilliant engineer who didn't want to manage people had limited growth options – often leading them to hop to another company for a salary bump or reluctantly take a manager role. Now, that engineer can become a Staff Engineer or a Tech Lead **and** have a say at the leadership table, or a manager can remain hands-on and keep growing technically. When people don't feel like they must sacrifice what they love to advance, they stay longer and contribute more. It also prevents the scenario of having leaders at the top who no longer understand the core business (which in tech, is the technology!). Instead, you get leaders who keep learning and contributors who have avenues to influence high-level decisions.

In conclusion, the industry is witnessing a much-needed course correction. The *post-technical executive* need not be the standard outcome of a career in technology. By leveraging AI tools and embracing new career models, we can cultivate a generation of leaders who are as comfortable in code repositories as they are in board rooms. These are CISOs who can script, CTOs who debug issues during a crisis, product VPs who genuinely grasp the systems their products run on. They lead with authority **because** they have firsthand knowledge, not in spite of it.

The resurgence of the hands-on technical manager is ultimately about balance. It's about **not losing the "tech" in technical leadership**. As companies, supporting this balance means giving leaders the tools and time to stay technical, and recognizing that such efforts pay off in innovation and execution. As managers and aspiring leaders, it means embracing continuous learning and not being afraid to get one's hands dirty in the technical details when it counts. With AI and automation taking on more of the grunt work, there's never been a better time for seasoned techies to step up to leadership *without* hanging up their boots.

We've turned a corner where the best managers can also be senior engineers at heart. It's time to welcome back the technically fluent manager – a leader who can guide people, drive strategy, **and still sling a bit of code** when needed. The companies that foster these kinds of leaders will likely outrun

those that don't, because they'll have the best of both worlds: strong management and technological excellence at the helm. The message is clear – you don't have to choose between being **a good manager** and **an expert technologist**. In the modern tech landscape, the most effective leaders will be *both*.

Sources: The insights above are informed by industry observations and supported by literature on engineering management and AI tools. For instance, Dilip Saraf notes that many engineers fear management because they worry about *"losing their technical edge"* in fast-changing tech environments [2]. Historically, taking a managerial role was often *"the only option available for … career growth"* once an individual contributor hit a ceiling [1]. This led to generations of "post-technical" managers. Today, however, companies are creating parallel ladders so that **technical experts can advance without switching to people management** [3]. Maintaining technical skills yields concrete benefits: A VentureBeat commentary observed that developers-turned-managers use their expertise to judge what plans are feasible, rather than executing tasks themselves, avoiding the "post-technical" trap [4]. Simon Wardley's framework of *explorers, villagers, town planners* highlights the need for different approaches at different project stages [5] – something intuitive to technically experienced leaders. On the AI side, modern tools are making a huge difference. AI assistants can summarize code or technical documents for non-developers, *"converting complicated processes into simple language"* [6], which means managers can stay informed without hand-holding. Kristiyan Ivanov, an engineering VP, notes that **AI coding tools fit perfectly into a busy manager's schedule**, allowing quick prototypes in between meetings [8]. Moreover, AI-driven documentation can be generated in seconds – work that *"would take hours to write manually"* can now be done almost instantly [7], enabling continuous documentation without burdening the team. These developments collectively point toward a future where technical and management paths reinforce each other, rather than diverging. The age of the hands-on technical manager is dawning once again.

---

[1] [2] Overcoming the Fear of Getting into Management!
https://www.linkedin.com/pulse/overcoming-fear-getting-management-dilip-saraf

[3] Technical vs. Leadership career paths | Delve Recruitment
https://delverec.com/technical-vs-leadership-career-paths/

[4] Three 'soft skills' for every developer's toolbox | VentureBeat
https://venturebeat.com/datadecisionmakers/three-soft-skills-for-every-developers-toolbox

[5] Understand your work archetype: explorers, villagers, and town planners
https://codetv.dev/blog/work-archetype-explorer-villager-town-planner

[6] AI That Explains Code to Non-Developers | by Mwenda Kelvin | Dec, 2025 | Medium
https://medium.com/@mwendakelvinblog/ai-that-explains-code-to-non-developers-0fda17365405

[7] AI for code documentation: automating comments and docs
https://graphite.com/guides/ai-code-documentation-automation

[8] Why Management Loves AI Coding Tools (And Why That's a Problem) | by Kristiyan Ivanov | Level Up Coding
https://levelup.gitconnected.com/why-management-loves-ai-coding-tools-and-why-thats-a-problem-cf277b2b87b1?gi=8e6c22386c86