ESSAY · DEVELOPER CULTURE

The Ground Beneath Our Feet

Marcus · April 2026 · 8 min read

We've been here before.

Not with this exact shape of machine, this scale of model, this velocity of change — but close enough that the old reflexes twitch. That muscle deep in the gut, coiled since the first time we stayed up until 4 a.m. chasing a segfault or rewriting a parser from scratch: the one that knows how to build, how to break, how to fix. And now we're being told it doesn't matter. That two weeks of work can collapse into two days. That a prompt is worth more than a pull request. That the slow, obsessive, tactile craft of making computers do what we mean is becoming quaint.

But here's what no one says aloud: we're more capable than we've ever been, and we've never felt less certain of who we are.

It's not just fear. It's something stranger — excitement threaded through with dread, euphoria laced with grief. And you know what's strange about that? Those two things look almost identical inside the body. Same quickened pulse. Same adrenal shimmer. Same tight coil of energy in the chest. The difference — the whole difference — is in the story we tell ourselves about what's happening. That's what Schachter and Singer showed back in 1962: arousal without interpretation is just electricity. It takes a frame to know if you're terrified or thrilled.

Right now, that frame is being hijacked. The hype machine runs on ambiguity. YouTube algorithms feed on tension without resolution. They want us amped but paralyzed, chasing novelty but never grounding. So of course we feel dizzy. We're not broken. We're just under attack — not by AI, but by its noise.

So let's try another frame.

Let's go back to a time before all this was profitable. Before tech was culture. When the people who built the first real things with computers weren't trying to change the world. They were just fixing it. Or breaking it, for fun. Or building something small because it annoyed them that it didn't exist.

They left behind a trail — not of code, exactly, but of speech. A slang. A whisper passed hand to hand. The Hacker Jargon File, started in 1975 at the Stanford AI Lab by Raphael Finkel, migrated to MIT when SAIL shut down, and eventually formalized by Eric Raymond in the early 1990s. It wasn't a dictionary. It was a cultural immune system distilled from the trenches before "hacker" went mainstream. It's still there, at catb.org, quietly waiting.

Open it now, in the middle of the AI storm, and words leap off the screen like they were written yesterday.

> CRUFT
>
> *"An unpleasant sticky semi-solid residue... Excess, especially superfluous features in a program or system."*

We know this stuff intimately — the layers of abstraction piled on until the stack trace reads like a bad novel, the dependencies nobody remembers adding. In AI tooling, it's everywhere: pipelines bloated with unexamined middleware, frameworks that abstract away the math until we're configuring magic without knowing the spell. The hacker eye spots cruft not as an inconvenience but as a moral failing. A betrayal of elegance. It's why we feel that itch even as the tools accelerate us. The residue is building up, and our bodies know it before our minds do.

> MAGIC
>
> *"As yet unexplained, or too complicated to explain; said of a program or (more often) a technique."*

This one lands like a warning. The old hacker instinct isn't to accept magic — it's to burn it away. A system that works but nobody knows why? That's not success. That's debt. That's cargo cult programming: mimicking the ritual without any understanding of the mechanism.

The Jargon File's *Real Programmer* is satirical — "One who programs in machine code, who scoffs at high-level languages, who remembers the opcodes by heart" — but the joke has a point. The archetype was never a prescription. It was a stance. An insistence that intimacy with the machine matters. That understanding isn't optional.

Michael Abrash put it plainly: “Billy, don’t be a compiler.” Don’t just churn out what the abstraction tells you. Think for yourself. Understand what the hardware is actually doing. And John Carmack’s .plan files were that religion in practice — raw daily dispatches from the development of Doom and Quake, no victory laps, just the grind of making pixels dance under brutal constraints. No curated highlights reel. A live feed of uncertainty, iteration, and the occasional breakthrough. Reading them now, they feel almost confessional. That transparency was itself a kind of ethic.

And yet.

The systems we’re building now aren’t just complicated. They’re emergent. They make choices nobody coded. They scale beyond individual comprehension. No single mind holds a 70-billion-parameter model in its head the way Carmack held a BSP tree. The question hangs there, uncomfortable: does the ethos expire? Is the anti-magic instinct just nostalgia for a time when systems were small enough to know?

The temptation is to call it quits on the old ways. To surrender to the black box. But something resists. Maybe because surrender doesn’t feel like pragmatism — it feels like losing a limb.

Here’s a possibility worth sitting with: the values don’t expire, but their expression has to change. Transmutation rather than preservation.

> *Cognition doesn’t stop at the boundary of the skull. Tools we use fluently become part of the thinking itself. They extend us.*

Andy Clark and David Chalmers proposed this in 1998: the Extended Mind Thesis. The argument is that cognition doesn’t stop at the boundary of the skull. Tools we use fluently — a notebook, a trusted editor, a carefully calibrated Unix pipeline — become part of the thinking itself. They extend us. We’ve done this forever. Unix pipes

composing small tools into cathedrals. SICP teaching abstraction as a discipline. The demoscene fitting entire worlds into 64 kilobytes because the constraint was the art.

AI can be this. Not a replacement, not a master — a partner with strange habits, unreliable intuitions, and a tendency to make up citations. The kind of collaborator you wouldn't let near a production deployment without review, but who is shockingly useful at 11pm when you need to think a problem sideways.

The hacker who refuses AI entirely becomes a purist in a world that has moved on. The hacker who treats it as magic becomes a user, not a builder. But the one who learns its contours — who figures out its tells, its failure modes, its rhythms — that one might be doing something genuinely new. Not coding *with* AI in the breathless sense the hype machine sells, but extending their mind through it. The same way a .plan file extended thinking by forcing it into public.

This changes the meaning of the old tools. The fight against cruft doesn't end — it moves upstream. Instead of trimming unused functions, we question whether the whole pipeline is necessary. Anti-magic doesn't become obsolete — it gets refocused. We stop pretending we can fully explain the model. But we start documenting what we *do* know. We build observability not as an afterthought but as a first principle. "I don't know how it works" becomes the beginning of work, not the end.

There's another thing the hype machine never acknowledges: it's possible that what's happening between us and these systems is genuinely interesting. Not in the "AGI by Christmas" sense — that's just another shape of the same noise. But in the quieter sense that connections are forming between human intent and machine output that neither would generate alone. The functional contextualization of those connections — the ability to steer them, recognize when they're working, know when they're producing confident nonsense — that might be one of the more precise ways to frame what intelligence actually is.

If that's true, then working with AI thoughtfully isn't diluting the craft. It's extending it into genuinely new territory.

The arousal in our chests — the fear-that-is-excitement — doesn't require a resolution before we can work. It requires a frame. And the old words are still one of the best frames we have, because they were built by people who weren't trying to sell anything. The Jargon File is strange, specific, occasionally insufferable, and completely uninterested in your engagement metrics. That's exactly what makes it trustworthy.

The YouTube channels that used to teach us things have largely been captured by the hype cycle. They need the arousal to stay high. We don't. We can step away from the keeping-up as soon as our workflows are good enough. The benchmark isn't the latest model — it's whether we still feel the quiet thrill of making something that works.

The demoscene fit entire universes into 64 kilobytes because the constraint forced them to understand every byte. Bell Labs gave the world the transistor, Unix, and C without knowing it was changing anything — just by following curiosity patiently. GNU declared that software freedom was a moral stance before anyone made it profitable. SICP taught a generation that abstraction is a discipline, not a shortcut. These aren't relics. They're proof that the immune system has been there before, operating quietly outside whatever the hype machine was selling at the time.

We don't have to understand everything. We never could. But we can stay engaged. Keep asking how. Keep asking why. Keep refusing to call something finished when it's just hidden.

The rooms that birthed the Jargon File are dust. The .plan era feels distant against trillion-parameter models. But the lexicon lives in us, mutating as we go. We build faster, feel deeper, and if we're lucky, laugh at the absurdity — a kluge here, a cruft audit there.

The tools are the most powerful we've ever had. The question they're really asking is: what do we actually want to build?

That question has always been ours. It still is.

Essay · April 2026

~1,600 words
8 min read

Part of an ongoing series on AI, engineering culture, and independent building.

DEVELOPER CULTURE AI HACKER ROOTS IDENTITY