Geoff White
Individual article

Add credit and caption

From Punched Cards to Prompts: Rediscovering Engineering Through AI

It's no secret that generative AI is reshaping the tech landscape at exponential speed. And among seasoned engineers, I've seen every kind of reaction from curiosity to outright denial.

There are basically a few responses that I've seen people have:

1. **F*** AI I'm not changing the way I do things.
2. **AI? It's just a fad.** There'll be something else next year.
3. **AI is going to kill us all!**
4. **I'm worried AI will take my job.**
5. **It sucks to have to learn a new set of tools, but... progress.**
6. **Can't wait for Skynet!**
7. **This AI stuff is like a force multiplier for my brain.**

All valid responses but only a couple lead you down a path of growth. **I chose the last one: seeing AI as a force multiplier for the human brain.**

The Engineer Who Stopped Coding — Then Started Again

I wrote my first programs on punched cards in FORTRAN IV. Over the decades, I moved through BASIC+, FORTH, numerous assembly languages, C, C++, and eventually Python. But sometime around 2014, I stopped doing serious programming. The younger engineers coming up were faster at everything — typing, coding, design.

My edge was in system troubleshooting the pattern recognition that comes only from decades of seeing how things break. I figured that would be my last "superpower."

Then came generative AI.

Around 2010, while at VMware, I'd written a tool called *esxplot* a Python 2.7 app that parsed massive ESXi hypervisor data sets and visualized them interactively. It proved more useful to Solutions Engineers and

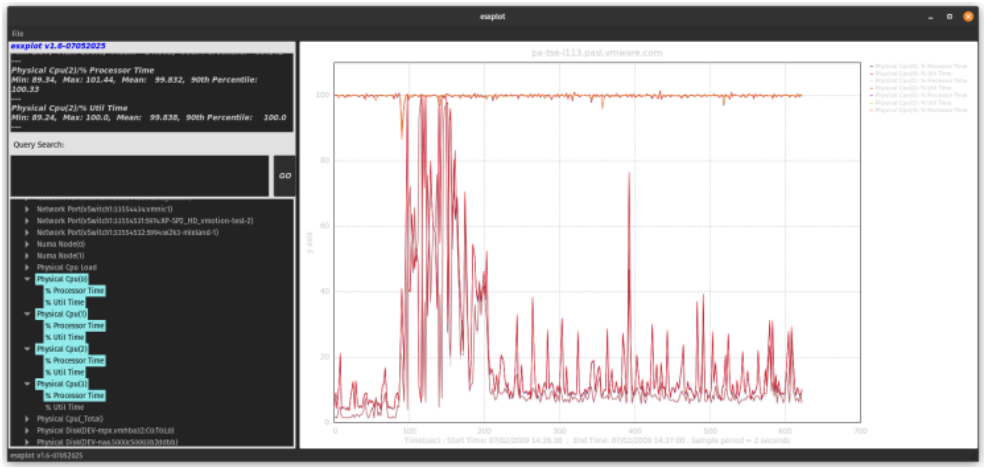
Professional Services than I'd expected. For years, I thought about writing a successor but never found the time.

So, as a challenge, I wondered: could AI help me resurrect *esxplot*?

I connected the old repo to an AI enabled IDE, Windsurf paired it with ChatGPT, and simply asked:

> "Can you upgrade this code to Python 3.5?"

It said it could. **Two hours later, the program was running on my Linux box dependencies updated, wxPython modernized, and the GUI working again.**



esxplot resurrected

But here's the thing: most of that two hours was spent by *me* examining the proposed changes, and hitting "Accept". I didn't even have to do the debugging, if the app crashed, I'd cut and past the stacktrace in the

chat window and the AI would locate the issue, find a fix, test it and integrate it, The AI did the heavy lift; I provided the discernment. That distinction matters.

That was my moment of clarity. I hadn't just upgraded a project **I had upgraded myself.**

The AI Suite That Rebooted My Engineering Practice

Since that day, I've leaned into mastering the tools that amplify my workflow. My short list:

- **Windsurf** and **Cursor** AI-powered IDEs that make coding iterative and conversational
- **ChatGPT, Claude, Gemini, Perplexity, DeepSeek** foundation models for ideation, debugging, translation and rapid research
- **Midjourney** and **DALL·E** for image generation and visualization
- **NotebookLM** – holds sets of evolving knowledge bases

If you want a structured path, what worked for me, I've found **Skillleap (now part of Futurepedia)** to be one of the few resources that doesn't waste your time. It offers:

- Over **30 AI courses and 1,000+ lessons**, regularly updated
- A **private professional community** for peer exchange and collaboration
- Concise, no-fluff micro-learning that gets you productive fast

And just to be clear, **this isn't a promo for Futurepedia.** I don't have any affiliation with them. I just find it *amazingly useful* , it's saved me a lot of time, frustration, and that low-level anguish we all feel when trying to separate real signal from the AI noise.

These tools aren't replacements. They're accelerators, turning an experienced engineer's intuition into executable velocity. They teach you "how to fish".

What I've Learned So Far

If you've built systems for decades, the most valuable thing you bring isn't syntax fluency, it's **context**. AI can write code, but it can't yet understand architecture, trade-offs, operational nuance or corporate intrigue, like a human who's lived through outages and redesigns.

That means the best engineers in the AI era will be the ones who *learn to direct AI*, not fear it.

Practical Advice for Senior Engineers Adopting AI

1. **Start with something you already care about.** Use AI to refactor or maintain, not rewrite from scratch.
2. **Review every suggestion.** Don't click "accept" without understanding what changed.
3. **Build an AI runbook.** Keep templates, prompts, and lessons in one place.
4. **Retain your mental model.** Use AI to probe and accelerate, not replace your reasoning.
5. **Be the bridge.** Mentor juniors or peers — helping others consolidate their skills strengthens yours.

The Takeaway

This isn't about keeping up with the next wave. It's about rediscovering software engineering, the curiosity, the craft, the reinvention.

The tools have changed. The mission hasn't.

Call to action:

I'm excited to continue the journey with my AI agents and assistants at my side. Some of the areas I'm looking forward to exploring are:

- Running OpenSource DeepSeek in Production - would really like to work on that.
- Local LLMs, running on your home lab
- SLMs - tiny "LLMs" that work in smartphones and devices.
- AI assisted Rust Programming
- Using AI tools to become a more prolific writer... knowledge transfer.
- AI education - helping my colleagues pivot.

If you've experimented with AI in your engineering workflow or if you're hesitant to begin, tell me:

What's worked for you? What's tripping you up?

Let's compare notes. The next evolution of our craft will be built together.