Before You "Vibe Code" Your Way Into Trouble: Essential Disciplines for AI Development Tools

By Eric E. Cohen, CPA

The pitch is irresistible: just describe what you want, and AI will build it for you. No coding experience required. Everyone's doing it. Don't get left behind.

But here's what they're not telling you in the breathless LinkedIn posts about AI coding agents: **powerful tools wrapped in friendly interfaces are still powerful … and potentially dangerous … tools.**

I've been thinking about this a lot lately. The analogy that keeps coming to mind? Those gummy dietary supplements my doctor recommended. They're candy-shaped medicine. Easy to take. A little too easy. Before you know it, you've had three instead of one, and suddenly you're dealing with consequences that potentially go way beyond a sugar rush.

AI coding tools – especially the new "agentic" ones that can reorganize your local files, commit to GitHub and restructure entire directories – work the same way. The interface says "just vibe with me," but the backend is running *rm -rf* commands on your file system. (That's Unix-speak for deleting stuff, recursively and forcefully.)

The Stories We're Not Sharing
While everyone's posting their wins, the quiet disasters are piling up:

- Working code replaced with broken iterations, no way to roll back.
- Entire project directories reorganized into oblivion.
- GitHub repos polluted with dozens of AI-generated commits that obscure the actual development history.
- Local files deleted because the AI "cleaned up" what it thought was redundant.

I have faced these personally.

The Disciplines We Actually Need
If you're experimenting with AI coding tools – and you should be – here are the non-negotiable practices to adopt first:

1. The 3-2-1 Backup Rule Still Applies Before letting any AI agent touch your files: three copies of your data, on two different media types, with one copy off-site. If you're not backing up, you're not experimenting – you're gambling.

2. Version Control Is Your Safety Net Learn about git (local) and Github (cloud). Git isn't optional anymore. Start with the basics. If you continue in a bigger way, expand on the disciple. Every meaningful change should be *committed*. Every AI-generated change should be in its own *branch*. Learn to read diffs before you merge. The AI is going to generate a lot of commits – make sure you understand what each one does.

3. Sandbox Everything First There's a reason Claude for Computer Use's reference implementation uses "Docker," a development sandbox and ChatGPT uses its own virtual machine. These protect your system. You should do the same. The tools themselves warn you to not use them from a top-level directory. Test AI tools on throwaway projects in isolated subdirectories before pointing them at anything that matters. Create a dummy directory structure. Let the AI make mistakes there. Learn what "clean up my project" actually means before you let it touch your business files.

4. Read Before You Run When the AI suggests running a command or script, read it**.** Understand it. Google the parts you don't recognize. "Trust but verify" isn't paranoia – it's professionalism. (Thank you, Massie family.)

5. Set Boundaries on Agent Autonomy Just because an AI *can* reorganize your entire codebase doesn't mean it should. Start with read-only access. Graduate to write access in limited directories. Full system access should be earned, not default.

6. Document Your Experiments Keep a log of what you're trying and why. When something breaks (not if, when), you'll need to retrace your steps. The AI won't remember what it did three sessions ago. The number of projects I started sitting on my drive that I have forgotten about, let alone remember how to use, is greater than one.

The Uncomfortable Truth
The pressure to "experiment or get left behind" is real. But here's the thing: **the people telling you to move fast aren't the ones who will help you recover your files.**

The most successful AI adopters aren't necessarily the ones who dive in headfirst. They're the ones who bring discipline to the chaos. They're the ones who understand that a friendly interface doesn't change the need for backups, version control and careful review.

For Parents and Business Leaders
If you're encouraging your team or your kids to experiment with these tools – and you should be – make sure they learn the fundamentals first:

- How to back up their work.
- How version control works.
- How to read what the AI is actually doing.
- When to say "let me review that first."

The best time to learn these disciplines is before you need them.

The Bottom Line
AI coding tools are transformative. They're also tools that can cause real damage when used carelessly. The solution isn't to avoid them; it's to bring the right disciplines to them.
Before you "vibe code" your next project, ask yourself: Do I have backups? Do I understand version control? Can I recover if this goes wrong?

If the answer to any of those is no, you're not ready to let an AI agent reorganize your file system. And that's okay. Take the time to build the guardrails first.

The technology will still be there when you're ready. Your files might not be.

What disciplines have you found essential when working with AI development tools? Have you had any close calls? Share your experiences; we all learn from each other's mistakes.