Vibe Coding - Style Over Substance

Jarrod Connolly

January 1, 2026

Abstract

AI can whisper elegant functions into a codebase, and that magic can hide a shallow understanding of the work. This article is about vibe coding, style over substance, and using AI as a sidekick rather than a replacement for craft. I draw a line from the sci-fi companions I grew up with to how I actually write software today.

Introduction

Picture this: you are knee-deep in code, and AI whispers the perfect function, elegant and efficient, almost magical. But what if that magic is just smoke and mirrors, a “vibe” that masks shallow understanding? I grew up devouring sci-fi novels where AI companions guided heroes through chaos. Those stories taught me to see AI as a trusty sidekick, not a crutch. Yet in today’s coding world, “vibe coding” is creeping in: style over substance, AI doing the heavy lifting without the human touch.

This is not a rant. It is my personal take, shaped by decades of fiction and real-world tinkering, on how AI should amplify our skills rather than replace them. I will unpack what vibe coding means, why it worries me, and how the sci-fi I grew up on points toward a better way of working with these tools.

What is Vibe Coding?

“Vibe coding” has become shorthand for using AI as the ultimate shortcut. Non-technical folks often see it as the evolution of no-code tools: describe an app, watch AI generate it. The pitch is democratization. Anyone can clone a social media platform or an e-commerce site without writing a line of code. A flood of generators and wrappers has appeared to capitalize on this, promising apps in minutes.

I dislike this take. It flips the script on what coding should be. It is all flash and no foundation. When AI generates entire projects, the result is often fragile, untested code that breaks under real-world pressure. The learning, the debugging, and the ownership that make coding meaningful get skipped. AI should assist the craft, not replace it. Otherwise we are building castles on sand.

Code Quality

When AI handles too much at once, code quality slips. Are people truly reviewing AI-generated pull requests? Often it is a quick glance, or skipped entirely. Reviewers trust the AI, or overlook odd patterns. And who ensures the code follows team style rules? AI might not match your guidelines, leading to

inconsistencies that make the codebase harder to navigate and update. Careful human checks keep things maintainable. Shortcuts undermine the foundation.

Security Concerns
In the rush to vibe code, we sometimes forget the basics of security. AI might produce code that looks great on the surface but overlooks vulnerabilities: weak authentication, unpatched libraries, exposed secrets. An app that crashes is a headache. A security breach is unrecoverable: leaked credentials, stolen data, shattered trust. Credential leaks are on the rise. Treating security as an afterthought in AI-assisted projects is a gamble nobody wins.

Eroding Creativity
Software engineering has always struck me as an art form. Creativity shines in solving novel challenges with code. I love seeing developers craft elegant solutions to complex problems, blending logic with imagination. With vibe coding, I worry we lose that. When AI handles the bulk of the work, the hands-on learning, debugging, and inventive thinking that build real expertise start to fade. It does not have to be inevitable. Keeping AI as a tool for exploration rather than wholesale execution preserves the art of coding.

Legal and Ethical Quandaries
If AI wrote the code, who is responsible for the bugs? The developer, the company, or the AI provider? What about insurance? Do policies cover AI-assisted projects, or do we need new frameworks? These questions are not theoretical anymore. As AI blurs the lines of ownership, human accountability needs to stay at the center.

The Clone Craze

Every day a new business pops up promising to clone any app in minutes. Like any emerging tech, these wrappers and generators are having their moment. Quick wins, shallow results. The excitement is real, but these clones rarely stand up to real user demands. Substance matters more than style, and style is all most of them have.

What Sci-Fi Taught Me

Sci-fi has a knack for inventing the future. I grew up immersed in these stories, learning about AI not as a cold machine but as a companion that amplifies human potential. Three books in particular shaped how I think about working with AI today.

The Hitchhiker’s Guide to the Galaxy
Douglas Adams gave us the Hitchhiker’s Guide: an electronic book packed with quirky knowledge on everything from galactic customs to towel etiquette. Arthur Dent interacts with it as a chatty companion, not cold tech. It has a dry, sarcastic wit, informative and irreverent in equal measure. It is an essential survival tool, but it never dominates the scene. It amplifies human curiosity and then gets out of the way. That is the dynamic I want from AI: helpful, clever, humbly subservient to the human journey.

The Diamond Age
Neal Stephenson’s Primer is an adaptive AI book that serves as a personalized teacher for a young girl named Nell. It adapts its lessons to her needs, her emotions, her growth. Characters interact with it like a wise mentor, even a surrogate parent. The Primer does not impose. It observes, adapts, and guides. That

vision of AI as a tailored assistant, one that educates without overwhelming, reinforced something I still believe: the best tools meet you where you are and help you grow at your own pace.

Snow Crash

Also by Stephenson, the Librarian is an AI research assistant that pulls and organizes information from vast databases. Hiro Protagonist queries it conversationally in the metaverse, and it responds with precise, context-aware data. There is a moment where Hiro shifts topics mid-conversation, and the Librarian handles it smoothly. I call these “conversation side quests,” and they are everything in my daily AI interactions: following a thread, branching off, coming back, never losing context. The Librarian is not a companion. It is a streamlined tool that blends machine precision with helpful dialogue. That is the model I reach for most often.

These three AIs share a common thread: they assist without replacing. They amplify the human at the center of the story. That is the standard I measure today’s tools against.

How I Actually Use AI

So what does this look like in practice? AI, used well, is a force multiplier. But it needs structure. Here is how I work with it day to day.

Research and Documentation

AI excels at digging through documentation: function usage, alternative approaches, version compatibility. I can chat through an API conversationally and uncover details buried in manuals. This saves hours of sifting through pages.

I also use AI as a study partner. I buy PDFs of textbooks, parse them into Markdown, and break them into chapters loaded into VS Code. Then I talk through each chapter with an LLM, working through exercises and examples. (I wrote about building a dedicated tool for this workflow in the HAL article.) The ability to fork into side quests (exploring a related concept, chasing a curiosity) keeps learning dynamic. This approach has led to some of my biggest leaps in understanding.

Writing and Maintaining Code

AI helps keep code comments current as logic evolves. It suggests updates so comments reflect why something exists, not just what it does.

For refactoring, AI provides insight into how changes might ripple across a project. It can pull in context from documentation or past conversations to suggest safer approaches. For boilerplate (class templates, test skeletons, repetitive patterns) AI acts like a custom code generator that remembers my style. These are not glamorous tasks, but they consume real time. Offloading them to AI lets me focus on the unique logic.

Unit tests benefit from the same treatment. AI suggests edge cases, thinks through coverage gaps, and keeps tests aligned with new functionality as code changes.

Algorithm and Design Discussions

When I am weighing tradeoffs (memory versus CPU, a library versus a custom implementation) AI lays out pros and cons. It often provides a working implementation I can profile and analyze. The back-and-forth sharpens my own understanding. The AI is not making the decision for me. It is giving me more information to make it myself.

Specialized Agents

I use targeted agents for different roles: QA, Security, Coding, Planning, Product Manager, Research. Each has specific rules, constraints, tool access, and sometimes different models suited to its purpose. They work alongside each other, providing specialized support without overlap. Agents also remind me to keep tests and documentation current during development, so maintenance feels natural rather than like a chore I will get to later.

Human Oversight

Reviewing AI-generated code requires a careful eye. I prefer a mix: AI flags obvious issues, but humans ensure the code aligns with intent and team standards. It is our code. We need to understand every part of it. Blind trust is the enemy of quality.

Agents need to be held accountable to team conventions: coding style, security practices, documentation expectations. Custom prompts and specific rules keep them aligned. Tailored to your guidelines, AI becomes a team player rather than a wildcard.

What I’ve Learned

I came into the AI era with a head full of sci-fi and a belief that tools should serve the craft. Working with these systems every day has only reinforced that.

The sci-fi AIs I grew up admiring (the Guide, the Primer, the Librarian) were not oracles. They did not do the work for the protagonist. They handed over information, offered context, suggested paths, and then stepped back. The human made the decisions. The human owned the outcome.

That is the relationship I aim for with AI today. Not “build this for me.” More like “help me understand this, so I can build it better.”

If you are using AI in your own work, keep the human at the center. Review the code. Understand the decisions. Own the result. The tools are getting better every month, but the craft is still yours.