An LLM Codegen Hero's Journey

2025-04-17 · 2358 words · 12 minutes · ❤️ 35

I have spent a lot of time since my blog post about my LLM workflow talking to folks about codegen and how to get started, get better, and why it is interesting.

There has been an incredible amount of energy and interest in this topic. I have received a ton of emails from people who are working to figure all of this out. I started to notice that many people are struggling to figure out how to start, and how it all fits together. Then I realized that I have been hacking on this process since 2023 and I have seen some shit. Lol.

I was talking about this with friends (Fisaconites's represent) and I sent this message in response to a thread about AI assisted agents, and editors:

> *if i were starting out, i don't know if it is helpful to jump right into the "agent" coders. It is annoying and weird. having walked a few people through this (successfully, and not successfully) I find that the "hero's journey" of starting with the Copilot, moving to the copy and paste from Claude web, to the Cursor/continue, to the fully automated "agents" seems to be a successful way to adopt these things.*

This lead me to start thinking a lot about the journey and how to get started using agentic coding:

> *The caveat is that this is largely for people with experience. If you don't have much dev experience, then fuck it - jump to the end.* ***Our brains are often ruined by the rules of the past.***

A journey of sight and sound

Your thoughtful guide: Harper. iPhone X, 6/10/2018

This is my journey. It is largely the path I took. I think you could speed run it if you were compelled. I don't think you need to follow every step, but I do think every step is additive.

Here are the steps:

Step 1: Get out of bed with wonder and optimism

Lol. Just kidding. Who has time for that? It may help, but the world is falling apart and all we got is codegen to distract us.

It does help to assume that these type of workflows could work and could be additive. If you hate LLMs and don't think it will work, then you will not be successful here. ¯_(ツ)_/¯

Step 2: Start with AI-assisted autocomplete

This is the real step one! You need to spend enough time in the IDE context to know how well you would work with intellisense, zed autocomplete, Copilot, etc. It gives you an idea of how the LLM is working - and prepares you for the stupid shit it will often recommend.

People seem to want to skip this step and just jump to the end. Then they are like "this LLM is a piece of shit and can't do anything right!" Which is not accurate, but also can be true. The magic is in the nuance. Or as I like to remember: *life is confusing*.

Step 3: Start using Copilot as more than autocomplete

Once you have a good process in place with the autocomplete and you are not mad *all* of the time, you can move on to the magic of talking to Copilot.

VS Code has a pane where you can Q&A with Copilot and it will help you with your code, etc. It is pretty cool. You can have a nice convo about your code, and it will be thoughtful and help you solve whatever query you asked.

However, using Copilot is like using a time machine to talk to ChatGPT in 2024. It isn't *that* great.

You will be wanting more.

Step 4: Move to copying and pasting code into Claude or ChatGPT

You start to satisfy your curiosity by pasting code into the browser based foundational model and asking "WHY CODE BROKE??" And then having LLM respond with a coherent and helpful response.

You will be AMAZED! The results are going to blow your mind. You are going to start to build lots of weird shit, and doing really fun things with code again. Mostly cuz it cut out the entire debugging process.

You can also do wild things like paste in a Python script and tell the LLM "make this into go" and it will just *make it into go*. You will start thinking "I wonder if I can one shot this."

Copilot will start to look like 2004 autocomplete. It is handy, but not really necessary.

This will lead you down a couple sub paths:

You will start to prefer one model cuz of vibes

This is the unfortunate first step towards the vibe in vibe coding. You will start to prefer how one of the big models talk to you. It is feelings tho. Kind of weird. You will find yourself thinking "I like how Claude makes me feel."

Many developers seem to like Claude. I use both, but mostly Claude for code related things. The vibe with Claude is just better.

> *You have to pay for them to get the good stuff. So many friends are like "This is a piece of shit" and then you find out they are using a free model that barely*

> *works. Lol. This was more of an issue when the free version was ChatGPT 3.5, but make sure you are using a capable model before you throw the entire premise out.*

You will start thinking about how to make things go faster

After copying and pasting code into Claude for a few weeks you are going to realize that this is annoying. You are going to start working through context packing, and trying to fit more of your code into the LLM context window.

You will experiment with repomix, repo2txt, and other code context tools. Just so that you can slam your entire codebase into the Claude context window. There is a chance that you will even start writing shell scripts (well Claude will write them) to help make this process easier.

This is a turning point.

Step 5: Use an AI enabled IDE (Cursor, Windsurf?)

Then a friend will say "why don't you just use Cursor?"

It will completely blow your mind. All the magic you just experienced by copying and pasting is now available in your IDE. It is faster, it is fun, and it is close to magic.

At this point you are paying for like 5 different LLMs - what is another $20 a month.

It works super well, and you feel way way more productive.

You will start playing with the agentic coding features built directly into the editors. It will *basically* work. But you can see a destination on the horizon that may be better.

Step 6: You start planning before you code

Suddenly you find yourself building out very robust specs, PRDs, and to-do docs that you can pipe into the IDEs agent, or into Claude web.

You have never "written" so much documentation. You start to use other LLMs to write more robust documentation. You are transposing docs from one context (PRD) to another ("Can you make this into prompts"). You start to use the LLM to design your codegen prompts.

You are saying the word "waterfall" with a lot less disdain. If you are old, you may be fondly remembering the late 90s and early 2000s and wonder "is this what Martin Fowler felt like before 2001?"

In the world of codegen: The spec is the godhead.

Step 7: You start playing with aider to enable quicker loops

At this point you are ready to start getting into the **good stuff**. The codegen previously required you to be involved, and paying attention. But it is 2025! Who wants to code with their fingers?

> *One other path that lots of friends are experimenting with is to code with your voice. To start instruct aider via a whisper client. It is hilarious and fun. MacWhisper is a very good tool for this locally. Aqua, and superwhisper are nice but cost more. They may use cloud services to do the inference. I prefer local.*

Trying out aider is a wild experience. You start it up, it instantiates itself into your project. You put your query directly into aider, and it just kind of does what you asked. It asks for permission to act, and gives you a framework to get things done, and then acts. It completes the task, and the commits to your repository. You no longer are so worried about one shotting tasks. you will just have aider do it in a few steps.

You start building out rulesets for the LLM to follow. You learn about the "Big Daddy" rule, or the "no deceptions" addition to your prompts. You start be really good at prompting the robot.

It works.

Eventually you don't even open up an IDE - you are just a terminal jockey now.

You spend your time watching the robot do your job.

Step 8: You lean all the way into agentic coding

You are now using an agent to code for you. The results are pretty good. There are a few times when you have no idea what's going on. But then you remember you can just ask it.

You start to experiment with Claude Code, Cline, etc. You are super happy to be able to use a reasoning model (deepseek!) and a coding model (Claude sonnet 3.7) together to start removing planning steps.

You are doing wild stuff like running 3-5 concurrent sessions. Just tabbing through terminals watching robots code.

You will start coding defensively:

- really hardcore test coverage
- thinking about formal verification
- using memory safe languages

- choosing languages based on compiler verbosity to help pack the context window

You will think long and hard about how to make sure that the thing you are building just gets built, safely without intervention.

You will spend **SO** much money on tokens. You will also use up all your GitHub action hours running all the wild tests that you are running to make sure that the code is built safely.

It feels good. You are not mad about not coding.

Step 9: You let the agent code, and you play video games

Suddenly, you are there. You are at the destination. Well, kind of - but you see where we are going. You start to worry about software jobs. Your friends are being laid off, and they can't get new jobs. It feels different this time around.

When you talk to your peers they think of you as a religious zealot cuz you are working within a different context than they are. You tell them "omg you have to try out agentic coding!" Maybe you add "I hate the word agentic" just to show that you have not drank 200 gallons of kool-aid. But you have. The world seems brighter cuz you are so productive with your code.

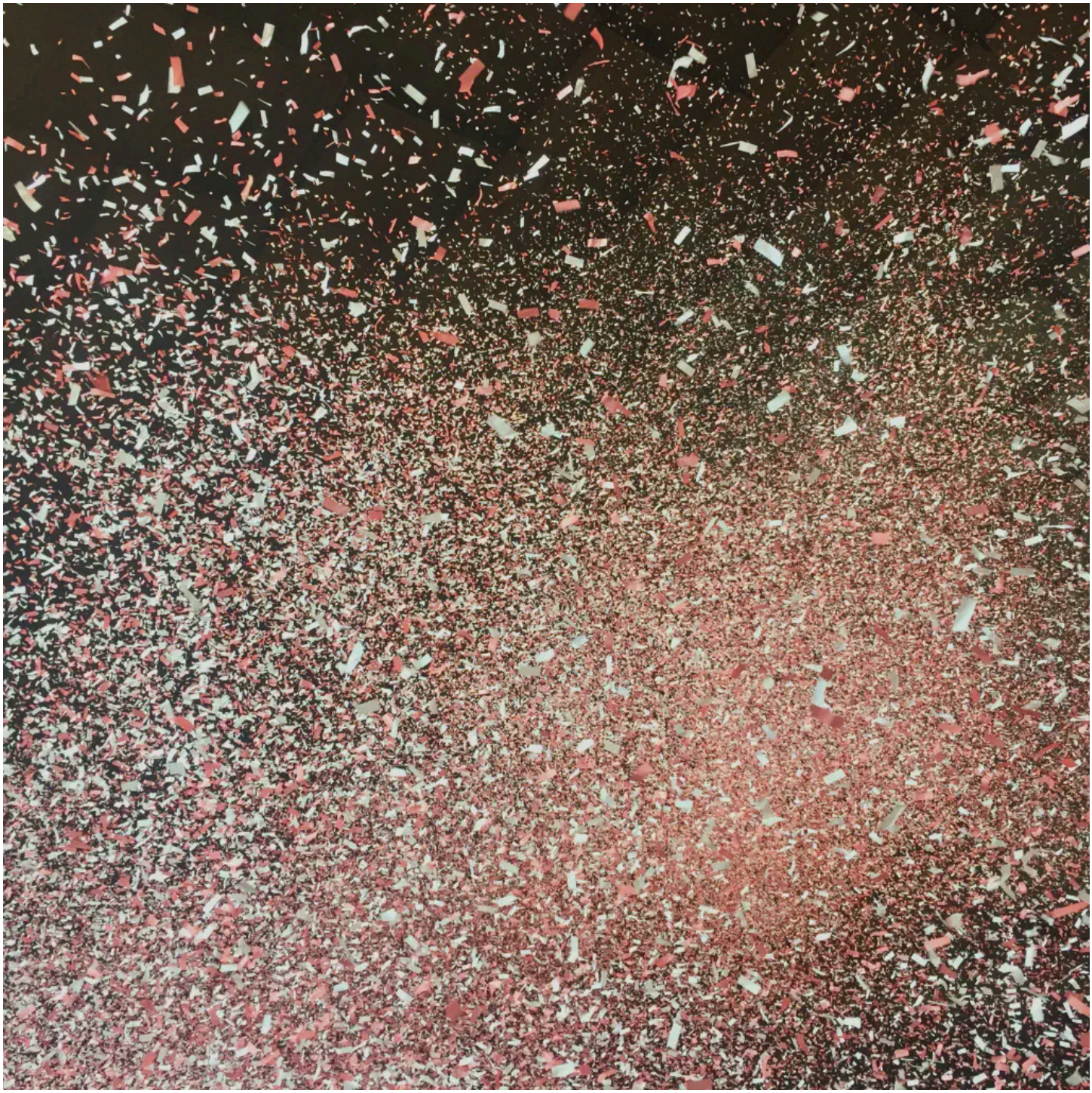
It doesn't matter. The paradigm has shifted. Kuhn could write a book about the confusion happening during this time.

Nobody can see this because they didn't go through the journey to get here. But those who have are agreeing and sharing their own tips around the journey, and debating the destination.

Now that you are knee-deep in letting robots do the work, you can really focus on all those gameboy games you have been wanting to play. There is a lot of downtime. And when the robot is done with a task, it will ask "should I continue" and you type **yes** and go back to Tetris.

Very strange. Unsettling, even.

The acceleration



Confetti at a Paul McCartney Concert at the Tokyo Dome. iPhone 6, 4/25/2015

I don’t know what will happen in the future. I am worried that people who are not working through this journey are not going to be attractive to employers. Which is kind of near-sighted, because, ultimately, we are talking about tooling, and automation.

When we were ramping up hiring in the past, we would often spread our queries well past our network, and past our tech stack. We would be a Python shop and interview people who didn’t know Python, and have never used Python. Our thought was that with a great engineer, we could work together to get them comfortable with Python. They would be additive even if they were not super comfortable with our stack. This worked out well for us. We hired incredible people who had never worked with our stack. Many times they brought such a different perspective that it elevated the entire team.

The same principles apply to AI-assisted development. When hiring talented developers who match your team's culture and show enthusiasm, their experience level with AI tools shouldn't be a deal-breaker. Not everyone needs to be an AI development expert from day one. Instead, guide them through the learning process at their own pace while they work alongside more experienced team members.

Eventually they will be the driver and will be successfully using these tools.

One other aspect I keep thinking about: Writing skills have become critical. While we've always valued strong communicators on tech teams for documentation and collaboration, it's doubly important now. Not only do you need to communicate with humans, you need to write clear, precise instructions for AI. Being able to craft effective prompts is becoming as vital as writing good code.

The leadership

I think all leaders and engineering managers need to dive deep into AI-assisted development, whether you're a believer or not. Here's why: The next generation of developers you'll be hiring will have learned to code primarily through AI tools and agents. This is what software engineering is becoming. We need to understand and adapt to this reality.

Us code boomers are not long for this world.

interesting note: i don't really use LLMs to help me write things. I imagine they would be good at it, but i find that i want my voice to be heard, and not normalized. Whereas my code needs to be normalized. interesting.

Thanks to Jesse, Sophie, the Vibez crew (Erik, Kanno, Braydon, and others), team 2389, and everyone else who gave me feedback on this post.

> This post was written 98% by a human.

Comments

Reply on Bluesky here

Dave Rutledge @daverutledge.com

I really enjoy reading these walkthroughs and updates. I'm basically a non-coder, but now I get to make my silly game ideas reality! dates.wiki and birthdays.wiki "today in history" timeline games

1 replies | 0 reposts | 2 likes

> Harper 🤴 @harper.lol
>
> It's so much fun
>
> 0 replies | 0 reposts | 1 likes

RocksThatCanRead @hardxcoded.bsky.social

just read the blog post, its like literary i just read the story of my life for last 6 months; I am taking the journey further though, through mcp tools, creating an agentic boutique soft dev agency, that helps to code from idea -> mvp -> prod. It so good to know that there more of us out there

1 replies | 0 reposts | 1 likes

> Harper 🤴 @harper.lol
>
> Awesome. I wanna see it!
>
> 1 replies | 0 reposts | 1 likes
>
> > RocksThatCanRead @hardxcoded.bsky.social
> >
> > will share when ready, stay tuned :)
> >
> > 0 replies | 0 reposts | 1 likes

Dave Rutledge @daverutledge.com

I'm curious what you would suggest for a 12-year-old just getting started on the idea of coding

1 replies | 0 reposts | 1 likes

> Harper 🤴 @harper.lol
>
> Hmm. Maybe bolt or lovable. They can get to seeing something really fast. V0 too. Do they want to code or do they want to make web stuff? If they want to code, then just hacking on something and using Claude as the teacher would be fun.
>
> 1 replies | 0 reposts | 0 likes
>
> > Dave Rutledge @daverutledge.com
> >
> > Thanks, I'll check those out! ...Probably 75% make stuff, 25% learn how to code. I'd love him to have a firm foundation of direct coding but I don't have any idea how to communicate job prospects for the field in ten years.
> >
> > 0 replies | 0 reposts | 0 likes