The 5 Step AI-Assisted Coding Loop: Don't Imitate Understand #8

Tony Alicea <hey@tonyalicea.dev>
Reply-To: Tony Alicea <hey@tonyalicea.dev>
To: <meldejesus22@gmail.com>

Mon, Jan 19 at 4:07 PM

Hello!

In this eighth issue of the "Don't Imitate Understand" newsletter you'll get:

- Book and course news then a discussion about a 5-step AI-assisted dev loop that I use daily.
- A bunch of interesting links!

New Free Book + Course News

- I am very pleased to announce that I am writing a book in public, and I'm making it available to read for free online. **The first 9 chapters are available right now!** The book is called Cascade Methodology: Software Development Practices for the Age of AI.

  It's a collection of best practices and new ideas for you and your team's software development lifecycle in the age of AI. AI has changed software dev, and tools like Agile and Scrum aren't designed to meet this new age.

  Cascade is an approach of my own design (based often on others' work as well) that you can integrate into your existing SDLC, or even replace entirely.

  So please give these first chapters a read! My email is open and I welcome any thoughts, as I will edit the book based on reader feedback.

- New modules are coming to my new Modern JavaScript Frameworks course and AI-assisted development course. You can still grab at the early-access price which ends in February, either for yourself or for your entire team.

The Spec → Plan → Decompose → Implement → Review Loop
Whatever AI model you may be using (or plan on using), there's a simple loop that I've found helps get controlled results that can be reviewed without overtaxing myself or a team.

That loop is:

Spec: A human-written spec (I prefer in markdown) of what the feature we are building. It should include why the feature is written and the context of the real world process the feature supports. That context ultimately produces more readable code (better function names, better pattern matching from code in the LLM's training set).

Plan: Ask the LLM to build a plan of how it will implement the spec and save that plan to a markdown file. Don't have it do anything yet. Have a human review the plan and adjust as needed.

Decompose: Take that plan and decompose it into *human-reviewable* smaller tasks. The biggest mistake is to ask the LLM to produce more code than you can comfortably review at a time.

You can ask the LLM to help you decompose the plan. It's ok to have separate decomposed Plan files. Whatever the case the decomposed tasks are better both for LLM context and for you and your team's sanity.

Implement: Ask the LLM to implement one decomposed task at a time.

Review: Review the LLM's code and test the feature yourself as you would normally.

This loop may not be the 100x, all agentic miracle process that you may read about online. But I can attest that it can work. You do need some other things though:

1. *Good context*: the spec, the decomposed plans, and any other information you would need to build it yourself.
2. *Good example code*: documentation (maybe via MCP), sample usages in the codebase, hand-built components for the LLM to use as a pattern to follow all do wonders.

How much faster will it make you? It depends on a lot of factors, especially how complex the code is that you have to review. But you *should* review LLM code before you make a PR.

I like acronyms, so let's pronounce SPDIR as “speedier”. If you like potatoes you can focus on the first steps and just call it “spud”. :)

Many experienced devs who I have talked to have settled on some form of this approach.

So be careful about listening to the “I produced thousands of lines of code in a weekend” crowd. They're trying for attention, they're not trying to save you stress. We are professional devs that live in the real world.

But with good context and good controls, you can get a lot done with LLM help.

Links

- Techniques for getting structured outputs out of LLMs
- Vercel is providing React Best Practices that you can add as a skill to your LLM.
- Speaking of LLM agent skills, here's the Agent Skills open standard.
- A long form post on how to write good specs for AI agents.

That's it for this eighth issue!

Happy coding!

Tony Alicea
Read the archive of this newsletter here.