Software Engineering Is Becoming Plan and Review — Louis Knight-Webb, Vibe Kanban

20 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=W76WOOYHLVY
https://www.youtube.com/watch?v=W76woOYHlvY

SUMMARY

Louie, founder of Vibe Canvas, discusses the evolving role of software engineers in the age of advanced AI, emphasizing that their work will increasingly focus on planning and reviewing rather than coding. He outlines the importance of strategic planning to optimize the use of AI tools, as well as the implications of longer execution times for coding agents on workflow management.

- *Software engineers will shift from coding to primarily planning and reviewing due to advancements in AI.*
- *AI tools like GitHub Copilot and Claude Code are reducing the time spent on writing code but increasing the need for planning and review.*
- *A plan-based approach to software development is more efficient, saving time in the review process.*
- *Different types of work (feature development vs. maintenance) require different approaches to planning and review.*
- *As AI agents become more capable, they will take longer to execute tasks, necessitating changes in how developers manage their workflow.*
- *The future of coding may involve AI performing QA tasks, potentially reducing the need for human oversight.*
- *Louie announced the shutdown of Vibe Canvas during the talk, citing challenges in monetization and market competition.*
- *The experience of running a startup has taught him valuable lessons about teamwork and the importance of hard work.*

[music] >> Is this mic on? Yes, this mic is on. How we doing? Woo! [__] fantastic. Yeah, let's go. All right. Um this I've This is So, the the title of this is is about planning and review, but I think the real point behind this is like basically what are we all going to do after AI continues to get really, really good.

Uh I'm Louie. I'm the founder of a startup called Vibe Canvas, and I also started the London chapter of AI Tinkers, uh which is great community um if you're in London looking for events. And you should listen to me because I have done some stuff like get on the SweeBench verified leaderboard ahead of Open AI. This is a couple of months old now, but anyway, you know, it's always nice to know that the people talking have done some research in the space.

Um the agenda for today, class, is we're going to uh I'm going to walk you through why I have arrived at the conclusion that basically all software engineers are going to do all day is plan and review stuff. And I'm going to talk about how to think about balancing that if that is what you do all day. Uh I'm going to talk about time horizon and how agents are getting uh running for longer and how that changes the behavior of the job. And

then at the end, we're going to shut my company down. And [snorts] we'll get on to that later on.

So, let's get started. Everything is plan and review. So, work that we software engineers do. Who's Everybody in here is a software engineer, right? Yes. Okay, most people. Um we plan stuff, we write code, we review code, and we review other people's code. Roughly, the work that we do. The ratios until GitHub Copilot hit the scene were roughly this for me. I know it depends on whether you work in a big company or a small company and things like that, but a lot of it was writing code and not very much of it compared to that was planning and reviewing code.

And what we see is over time with things like the first version of GitHub Copilot, that basically the writing code part starts to shrink. So, you know, ChatGPT arrived, suddenly, you know, you can like generate functions and paste them in, and then Cursor, not Cursor today, but like the original version of Cursor arrives, and then it's like able to complete a whole page of code. Um and then you get Claude Code, and it's like, "Wow, you know, I actually am not really doing much code writing anymore."

Um so, it kind of poses an interesting question though, which is what, you know, say we were spending 4 hours a day coding before, does that mean I now get 4 hours back if I'm not doing any actual coding? The answer, of course, is no. It has displaced work. Work that you or time that was previously spent doing the coding has moved. It has moved to planning and reviewing. I think it is an accelerant. You're probably getting more done in the day, but it's probably like, you know, you get uh 20 minutes back for every half an hour that you were, you know, spending coding, and some other time has gone to planning and reviewing.

So, I want to talk about that. Like, what is this new way of working? And I think there's there's fundamentally two approaches that people take. I'm not going to get too specific about, you know, I don't know, specs or uh Playwright MCPs or things like that. I think there's tons of fascinating talks about that at this event. I just want to kind of conceptually ground what we're talking about here today. So, the first way is the plan-based approach. Um this is where you spend a lot of time up front planning the work that you want a coding agent to do. So, the smells of whether you are doing this type of work would probably look like you're writing a very comprehensive plan doc, markdown file. You're maybe using like one of these spec frameworks.

You're uh interrogating. So, you know, I've seen some cool stuff where the model asks you questions repeatedly until it's like completely exhausted all possible questions it could have about what the work is that needs to be done. The benefits of this, of course, are that you basically spend less time reviewing that work because you have invested time up front uh eliminating edge cases, giving the the the models as much information as possible about the work you're trying to do. The outcome of that will be that it the models less likely [__] up, and you're going to get like better, better outputs, and probably, you know, fewer rounds of review. The downside is you have to spend more time planning, but, you know, that's just obvious, isn't it?

The other way of doing this, uh the the other big way of working with AI is you don't define a very detailed plan, but instead, uh you let it, you know, run, and then you you end up spending more time reviewing that work. So, you know, benefits of this, you can just YOLO something, be like, "Ah, let's add a contact form to the

webpage." And then, you know, the the the payback you have to do is like you're going to go back and forth a few times correcting the styles, figuring things out. Um I would say if you think about the valuable thing being your human time, and you have a choice, you always want to be doing the first of these behaviors, the planning the planning approach, basically, because it will save you a lot of time. It is very time-consuming to have to switch back and forth with an agent that is like giving you some half-delivered work, and you're constantly having to review it.

I think another way of breaking down the modes of work, where one is plan and one is review, is to think about the type of thing that you're working on. And I think feature development is actually very different from migrations and uh maintenance work. So, and front end is very different from back ends. I was I was trying to kind of think about this before the talk, and this is the matrix I've come up with, where basically, if you're working on uh the front end and you're doing feature development, it's basically impossible to kind of really spec everything out. There's so many edge cases and you know, front end is very stateful. Uh there's like interactions, animations, styles, there's functionality. And so, personally, like I find it much better to kind of be in the loop with a coding agent. So, the second uh one of those behaviors that we talked about. Uh but for everything else, I think it really is possible to be plan-heavy. So, back-end feature development, you can almost do test-driven development. And for anything like refactoring and migration-based, you certainly uh you certainly can be doing that, and you shouldn't be in the loop with any of that work at all, really. That should all be kind of test-driven development.

So, the I guess if you had to distill that long, meandering spiel into a sentence, it would be spending 5 minutes of planning saves you 30 minutes of reviewing AI-generated code. And that's basically the takeaway. Uh the other thing that I think is kind of interesting to consider is is how things are running for longer. So, as coding agents become more capable, so as models get better, as tool calling improves, you go from calling a very small set of tools to now, you know, the the coding CLIs can call a a huge range of tools and do testing and things like that.

The outcome of that is that every time you send off a prompt, you are waiting longer before it comes back to you and says, "Hey, Louie, it's time for you to do something." So, to illustrate this, I mean, you you know, like think back to GitHub Copilot. It completes a single line of code, and it takes seconds. Then you have like the original Cursor completing a single file, and that runs for, you know, 30 seconds. And then you have uh Claude Code, which, you know, last year would run for maybe a minute or two, and this year I've I've been, you know, getting some pretty good results with 5- or 10-minute executions. And that is going to continue because basically, we've gone from uh you asking the AI to do something and it just responds to the AI running a type checker to the AI testing its change. I mean, this is like the frontier of things. And these things take increasing amounts of time. Just returning the code was really quick. Running the type checker is a bit slower than that.

Running Playwright MCP is an order of magnitude slower than any of those things. Um but it's worth doing because, you know, what you're trying to maximize for, again, is how much time you are spend or minimize, rather, is how much time you are spending working with the agent. So, you know, if you can get higher accuracy by waiting longer, um that is a a worthwhile trade-off. And this is like where the frontier probably is is like, you know, if I had to forecast where we'd be at in 9 months' time, I would say basically, AI starts to be able to QA

front-end work, and that's going to be a huge breakthrough. You see some cool demos of this on Twitter with, you know, Chrome or Playwright MCP clicking around on stuff. The reality is I haven't met a single person who actually does this in their in their mainstream development.

But I'm really excited for it, and I think it will be the next major breakthrough, where essentially, most of the back and forth that you do with a model is going to just be done by the model itself because it'll be able to actually run your project, click around, and find the bugs, and make sure it's done it. But this poses an interesting question, which is like, what happens when the average time that an agent is running for exceeds, say 5 minutes. Cuz I think 5 minutes is roughly the time when you can like sit there and wait for something, watch the logs, probably more realistically like browse Twitter, something like that.

And when we cross that 5-minute mark, you have to change your behavior. You know, imagine these things are going to take 20 minutes to run. You're not going to sit there for 20 minutes watching agent logs. You're going to have to think about coding and all the job of being a software developer in a very very different way. Um this is something I'm sure everybody's seen, which is, you know, this kind of terminal maxing thing where you you basically parallelism, right?

You run multiple of these things at once. Say each of them take 10 minutes to run. So, the way you get around the waiting problem is you have multiple of them on the go at any given time. So, as soon as you finished, say reviewing one piece of work, another has finished and you can move on to that. And that's basically what we started working on. So, this is uh this is the the the project we started about a year ago called Vibe Kanban. And uh essentially it started as as an attempt to make it possible to paralyze agents very easily.

Um we built some cool stuff. There is uh a sidebar where you can create multiple workspaces that run any coding agent, like Codex, Cool Code, things like that. Uh when you want to review the code, you get the diffs. If you want to comment on something, you can do it just kind of like how GitHub does. If you want to preview something or, you know, click on something and kind of be like, "Ah, actually make this a bit bigger or that a bit smaller." You can do that, too.

And you probably have seen all of this stuff before, but you may not have seen this stuff started as early as June 14th, 2025. We did it first, I swear. Okay, so the considerations. I think like what so so so human behavior is going to change because agents are going to cross this like 5-minute threshold. Um and who knows, that may continue. You may end up crossing an hour threshold. So, we need new interfaces to make this job awesome. Cuz if you try and do it using the existing tools, it kind of sucks.

You have to jump around reviewing code in one thing, previewing things in another. Um if I had a wish list for what I would want the ultimate coding agent tool for software developers to look like, it basically embraced the fact that I have to be a manager of multiple streams of work at any given time, which is not something most software developers have had to do. They've just been able to like lock in in a in a deep way to one piece of work.

So, it's all about kind of I I put focus maxing. I don't know if that's a word. I'm coining it. You heard it here first. Um but it should embrace the fact that you can't pull humans out of something and back into something else every 30 seconds cuz it just fries their brain and it's not it's no way to live. Um so, you know, it it should be built around, you know, getting the most out of the human so that an agent can run for as long as possible and then yield back to the human rather than encouraging patterns where you're constantly jumping in and out and in and out of needing to get back into the context of what a particular agent is doing.

It should help you write tasks and plan things, obviously. It should help you QA work because that's what a lot of the human's work is going to be. And it should help you do code review. I think obviously code review, you know, a lot of it's being done with AI, but very few companies with money on the line are actually going to ship stuff that's fully Vibe coded without actually checking the code. So, reading code is probably something most people in this room are going to still have to do.

And then shepherding the change until it's deployed, which is kind of a new emerging one. So, you know, at its simplest form, it's like monitor GitHub pull requests and just look for comments and kind of be reactionary to those automatically. A lot of the admin involved in getting something from I finished the task to I've deployed the task is just literally like, you know, following comments and and reacting to them. So, this I wrote I wrote this talk I I I submitted this talk a few a few weeks ago.

And on Tuesday, I decided to shut the company down. So, [snorts] I had a whole basically there's a whole part of this talk which was just me telling you more about Vibe Kanban and trying to sell it to you, but that's not going to happen anymore because now the company's shutting down. So, what I thought I'd do instead is we can actually shut the company down together. I haven't I haven't actually announced it yet. So, >> [applause] >> Okay, and we're going to do it using Vibe Kanban, of course. It has to be, you know. Okay, so uh please add a blog post to the website with this content.

Uh >> [groaning and snorts] >> and I've pre-written the uh you know, the weepy the weepy note. Okay, so we've got Vibe Kanban website. All right, that's going to go and do it. I can give you a little tour of this thing as well. So, it's running a setup script. What it's created a Git work tree. It has run a setup script in the work tree to install the dependencies for our website. And once that's done, it proceeds to uh run whatever agent you've selected. I use uh Codex most of the time, but it supports eight of the most popular ones.

Um and it's going to go ahead and try and figure out how to do that. Um what else is cool? Are you sad? Am I sad? I think I've just done so much thinking about it. I'm kind of relieved. Like you run a you run a company for for a few years and you have this like enormous responsibility to kind of, you know, you have staff and investors and all this stuff and and I don't know, I feel like a kind of weight has been lifted almost. Um we can talk through like why we're shutting it down as well. We have we have lots of you We have 30,000 monthly active users and and 25,000 stars on GitHub. And actually the project will continue non-commercially. Um and we're already pushing changes even though we're we're shutting things down, but it's actually very difficult uh to make money in the current environment. Uh the everybody who is making money is doing two things.

They're selling to enterprise and they're reselling tokens. And we were doing neither of those things. We're not a coding agent. We have a button that helps you run something in Codex or or Cool Code. And so, people can we we have a subscription. People would spend like $30 with us and then press a button that helps them spend $3,000 with Codex. It's just like not sustainable. Um and all of our users are like individuals, startups, uh smaller companies. And we could have done the work, I think, to address that and kind of move up into the enterprise, but you know, I don't know. It's uh it's a kind of it's it's a mature market at this point and it's no fun playing for eighth place.

So, we decided to shut things down. Uh okay. Blog post is ready. Let's see if it works. So, we've got the live preview feature, obviously. Let's wait for it to compile. Okay. That looks good. So, we can go ahead, open a pull request. Uh uncommitted changes. Oh. Uh Don't know what that's about.

Uh Are we finding this out before your investors? Oh, [__] >> [snorts] >> Uh you're not. Don't worry. Uh they most of Well, actually some of them I need to I need to call them after this. That's a really good point. I'm fully committed to shutting this down live on stage. Okay. All right, it's done. That's going to go through Cloudflare's CDN.

>> [applause] >> Thank you. Um I think we've got time for one or two questions. I don't know if anybody in the audience has a a one 1 minute 45. Just >> Steve? What's next? Uh take some time off, start another company. My co-founders going to join the lab. And most of the team have already found good jobs at Agent Labs, things like that. Yeah. Any other questions? [snorts] Overall feeling when you look back is positive or negative Oh, yeah. No, I wouldn't I wouldn't do anything differently. I think like uh it it it it was like the most interesting thing I've worked on and uh is right at the cutting edge of like agents and all of this stuff. I think certainly I don't know. It's increased my value as a human by doing this, so I would do it all again. Yeah.

Go for it. What are the most valuable things you've learned from a few years of running a company? Uh most valuable things I mean, just work with great people. You know, we went through several rounds of the team uh and the team we ended up with was like phenomenally better. No no offense to previous team, but that's probably how to, you know, I think and and hard work as well. I think like it took us a while to learn like what hard work really was. Um and you get to a point that's like you're sitting there at midnight with the, you know, the team in the office on a Saturday and everybody's kind of motivated and that's yeah, it takes you a while to kind of figure out how to get to that point. And once you feel it, you know, kind of what that's like. It's difficult until you get there, I think.

Yeah. Uh 12 seconds. No? Okay. >> Looking back to the past, what would you change? What would I change? And I'd I'd hire somebody who's really good at selling to Enterprise. All right, thank you very much. >> [applause] >> Cheers. >> [music]