A note on using AI to write code for you or debug your code

Question from a recent Lyon graduate who now works as a software engineer, who asked for my recommendation regarding use of AI:

"I was wondering if you could give me any guidance. We are changing the way that we store stuff in our warehouse. They asked me if it would be possible to run some code with an AI to see what it would say. I asked ChatGPT and it gave me a pretty simple solution which involved getting the size, frequency grabbed, etc."

What do you think? Let's hear your answers:

1. Should a software engineer use AI to get ideas or improve his code?
2. Are there any potential issues with this approach?
3. Do you think AI will change software engineering, and how?
4. Should you use AI to let it help you learn in technical classes?

My answers (short version):

1. He should only use AI if he does not care about the result - if he does not depend on it, and if he has ample time to waste.
2. AI lies unpredictably, often delivers misguided output, makes you forget stuff you already knew, and gives you a false sense of security. It's like having a slave who really wants to help but is incapable of having even a single thought of his own.
3. It has already changed software engineering though there are no good studies yet, only a lot of calls for research. The pace of development of these principally intransparent tools is too fast.
4. No unless you have a lot of time to waste and/or already know your stuff really well (and even then, it's dangerous).

Here is my very long answer to the guy for the record:

Use AI for things you really don't care about, and if you don't care how much time you waste, because while the code it produces can be debugged, you will not know when and where the AI "hallucinates" i.e. spews nonsense. As a result, it may be hard to debug since you cannot talk to it about its own mistakes as you would to a coworker. AI aims to please you at any

cost and has no insight or understanding of what it does at all. Syntax errors are less likely, version and dependency errors are frequent (since it doesn't test its own code). Logic errors are brutal and hard to find and you may spend precious time dealing with the AI that you could have used to learn something new or test or create yourself (which would have taught you something). If you keep using it for things that are easy, you'll forget what you once knew. It's worse than copy-and-paste because it will always try to give you the whole thing, and copy-and-paste programming at least forces you to think the code through that you copied (ideally anyway). It is probably OK (haven't tried it) when it comes to web stuff since web stuff is easy. But it'll likely fail when you try it on anything that's complex (of course it has got better, too, over the past year, but only marginally, in this regard). In terms of raw code, its solutions are often not as easy as they could be, use the wrong data structures, have no concern for performance, and generally will mess with your own style - without offering anything in return (like a better or different style, which would happen if you worked with a human master programmer). Slight changes here depending on the language: C/C++ have lots of issues, Python probably the best (most public code examples I presume), SQL and R are pretty good (because they're simple and well-documented).

Having said that, I use it a lot (as I say in the talk) but only for stuff I don't care about very much, and also I am often not a slave to deadlines like you, and I have decades more experience to cut through the BS that I'm given. I use it when I'm lazy in the full knowledge that I may have to start all over, and to give me an alternative (which sometimes works), or sometimes because I'm lonely (unlike you I have nobody who programs with or alongside me). When AI teaches me something new, I have to re-learn it just as if I had heard it in a lecture or in a video (and check carefully).

I have spoken about this at length at Oklahoma University last March (link).

Some recent evidence in a home debugging story:

I set up a new workstation at home with a Linux distro that I had not used before. After installing Emacs from scratch, I got the error message: `"Invalid function: org-assert-version"`, which relates to a function in the Org-mode package and disabled many other commands at once. ChatGPT's advice was lengthy, involved re-installing software packages and peppering the Emacs configuration file with new functions. I put all of these suggestions to work since I didn't want to think about it - I just wanted to be done with (after many hours of installing things, late at night). The next morning I did not open the AI and just looked at the files that Emacs

complained about, and that coud not be processed. I immediately saw that there was an empty line where Emacs did not expect to find one (Org-roam expects to find an ID in the first line of the file). I had inadvertently created that empty line earlier when editing something. This is the old way - thinking about errors, getting frustrated, taking the error to bed and mulling it over, sometimes for hours, while doing something else in the meantime. Then, always, the answer "presents itself", or rather, it is generated by your wetware, your beautiful, God-given brain.

Short summary: At your stage, using AI is a waste of time at best, and a crime against your ability to learn at worst. Learning never comes without pain and (temporary) desperation. AI is like a pill but one that only works some of the time, and you'll never know when. Instead: join Lyon's Programming Student Club and experience the pain of not knowing first hand every week!

You don't have to believe me - experiment with it on your own but make sure that you don't get addicted to the convenience of AI.

Will you be punished for using AI in my class?

Not directly because nobody can tell if you used AI or not but indirectly by turning in suboptimal results, by learning less, and by having less time for other, more productive activities.

Are there any data on this?

Not much on coding per se but a recent (15 July), substantive, long (59 p) paper titled "Generative AI Can Harm Learning"), based on a very carefully conducted field experiment with a large (1000) sample of high school students concluded: "Our results suggest that students attempt to use [AI] as a "crutch" during practice problem sessions, and when successful, perform worse on their own. Thus, to maintain long-term productivity, we must be cautious when deploying generative AI to ensure humans continue to learn critical skills." (Bastani et al, 2024).

Here are two recent accounts from coders: Schluntz / Carlini

References

Bastani, Hamsa and Bastani, Osbert and Sungu, Alp and Ge, Haosen and Kabakcı, Özge and Mariman, Rei, Generative AI Can Harm Learning (July 15, 2024). Available at ssrn.com.

Carlini, How I Use "AI" (August 1, 2024). Available at carlini.com.

Schluntz, Replacing my Right Hand with AI (July 30, 2024). Available at erikschluntz.com.