I hesitate a bit to write about this, feels like I'm just jumping on the AI hype train. But something REALLY important is going on, and I want to talk about it.

I remember a friend telling me how AI was making incredible progress and might actually make human programming obsolete. I really thought it sounded like nonsense. I had all kinds of counter-arguments lined up. I don't think anything anyone could say or write would have convinced me at the time. Until one day I actually started interacting with GPT and seeing for myself. That kind of ch

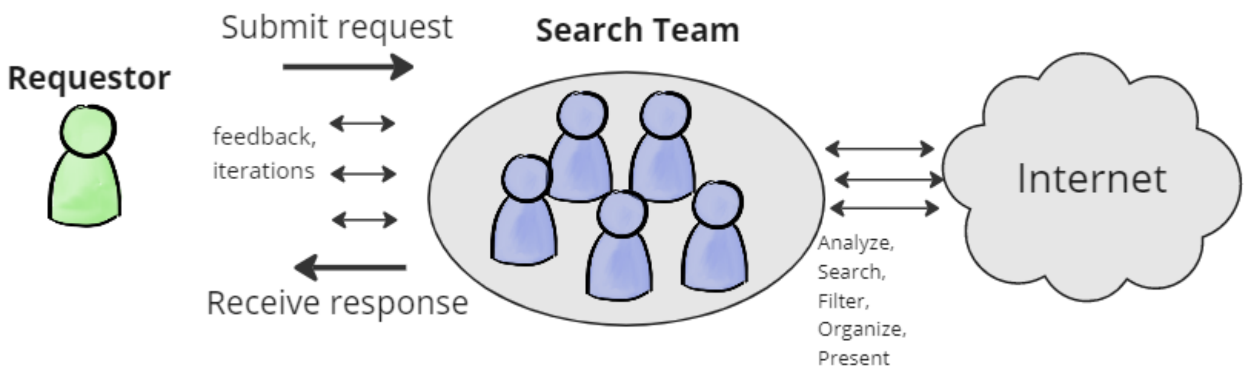
'erything.

There's so much hype right now about all of this so I understand some people might be feeling skeptical or cynical about it. If so, I recommend just trying it out. For example here: https://openai.com/product/gpt-4. Maybe there's a later version by the time you read this.

Ask it to do some of the work you usually do, whether it is creative or technical. It is far from perfect, but you will likely be surprised and more open to some of the things I'm talking about in this article.

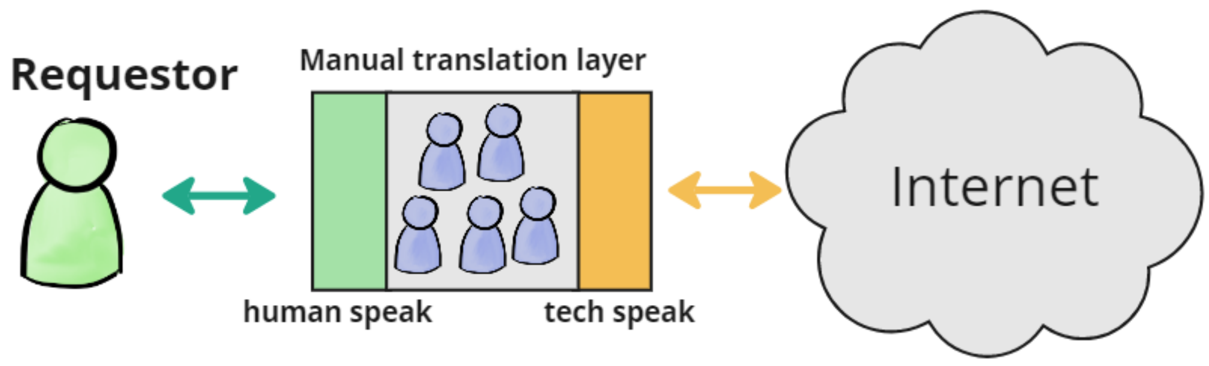
A silly thought experiment

Let's do a silly thought experiment: Imagine if the Internet was a new thing. Imagine you can find anything you want on the Internet, but doing so is complicated and requires expert knowledge and coding. So you have Search teams, people that are experts at searching the Internet.

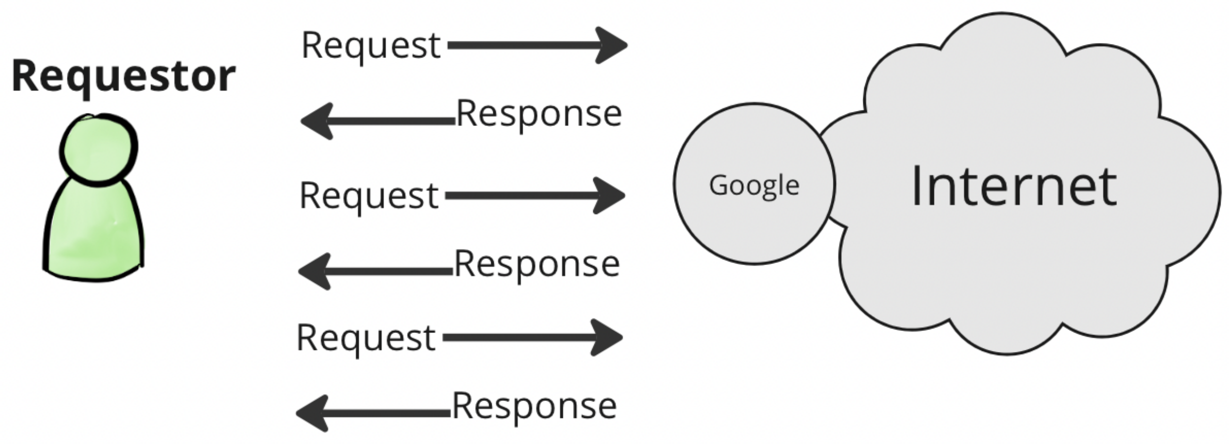


They receive search requests from other people in the company, let's call them Requestors. Each search takes a few days or even weeks sometimes. The team spends time discussing with the Requestor and understanding what information is needed and why, then they need to figure out which sources to consult, write carefully crafted queries, handle errors, organize and verify and filter the results, and then present it in a useful way. For more complex searches, a feedback loop is needed with multiple iterations of search results before the team arrives at a final search result. Each such loop could take days, so the final result could take weeks.

In this model, the Search team is essentially a **Manual Translation Layer** between the Requestor and the Internet. Simply put, the Requestor speaks Human-speak, while the Internet speaks Tech-speak, so someone needs to do the translation - well-paid experts who speak both languages.

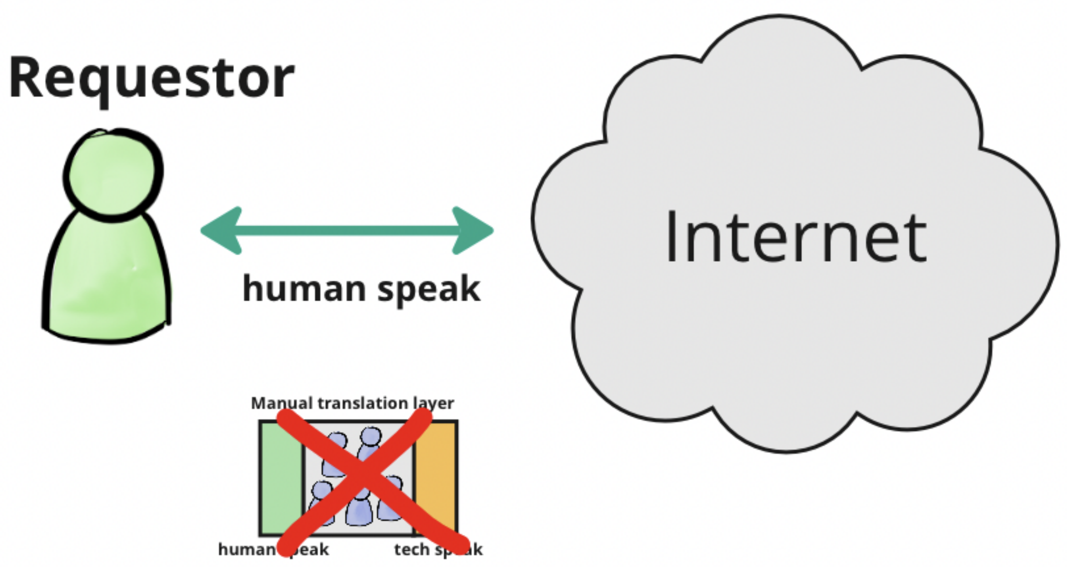


Now imagine if search gets easier and easier, gradually approaching what we have today - where anyone can search the Internet and get useful results within seconds, without any expert knowledge. Sure, the query might not be perfectly crafted, the initial answers might not be exactly what you were looking for. But the feedback loop is just a few seconds long, so you can just keep adjusting your query and retrying until you find what you need. Sounds familiar right? That's the world we're all used to living in now.



Where did that hypothetical Search team go? They are gone! All the work they were doing has been automated by software. The Requestor has a good enough tool that they don't need help from a team.

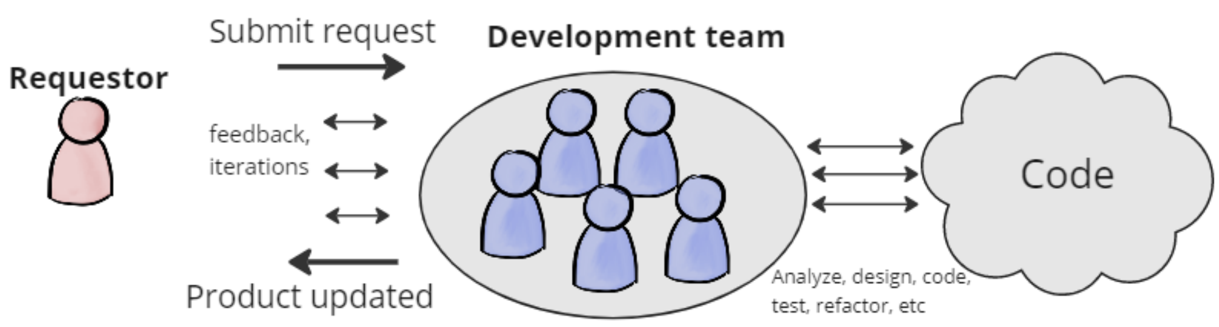
Simply put, if the Internet speaks Human-speak in the first place, then the manual translation layer is no longer needed. That saves us a ton of money as well as time. Technically, the translation layer is still there behind the scenes, but it is fully automated and lightning fast, so from the Requestor perspective there is no need for expensive time-consuming manual translation.



Sure, sometimes there are exceptions. Sometimes you DO need a Search team, although you might call it a Research team or something - a team that goes beyond just google searches, they actually dig up research papers, run surveys, interview experts, etc. But that's the exception. For day-to-day Internet searching no human intermediary is needed - anyone who wants to find something can do it themselves using google and similar tools.

What is a development team?

Now replace the word "Search team" with "Development team", and replace "Internet" with "Code".

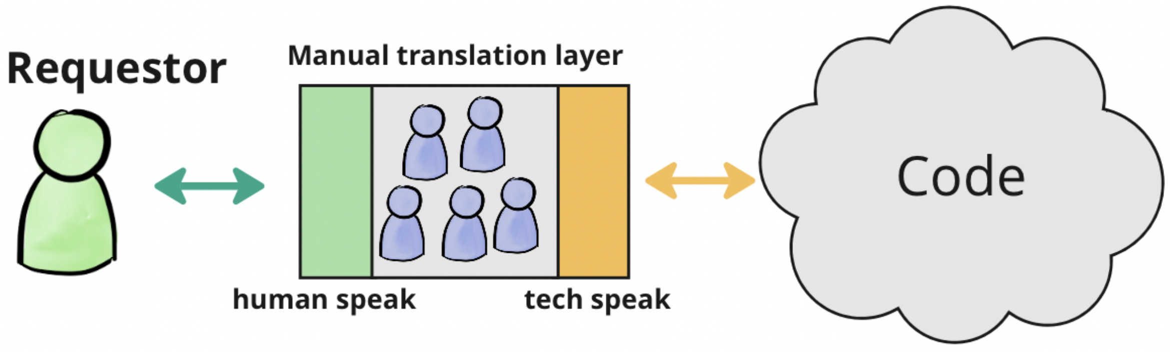


Looks pretty familiar doesn't it? This is very similar to the Search team example.

Again, I'm using "Requestor" as an abstract term for people that want stuff from the team, for example customers, users, product managers/owners, and other stakeholders. You can also think of "Requestor" as the person who hired the team and pays their salaries, and therefore gets to decide what the team should work on.

Now think about it. The Development teams talk to humans (what I'm calling the Requestor) to understand what needs to be developed and why. Then they turn around and write code to make it happen. Ideally the team does this iteratively, so there is some back-and-forth as the team iterates on the code and asks for feedback.

The Development team is a manual translation layer between the person who wants a problem solved, and the code that solves the problem.



And that, of course, begs the question: Do we need a manual translation layer for software development? Or bluntly put, do we need Development teams?

Do we need Development teams?

I have decades of experience working in the software development field in a bunch of different roles - Developer, Leader, Coach, Designer, Entrepreneur. If you asked me this question any time up until mid-2022, I would say "Yes, humans are definitely needed for software development, the manual translation layer is unavoidable".

Then ChatGPT showed up in late 2022. I tried using it to write code, and even to design products. It was not perfect. But I was still mind blown! After that my answer would be "Um, Yes humans are needed for software development, but tools like this will be an essential part of their job, and will make them super productive. For simpler programming tasks the AI might be able to write the whole program without human developers, but for larger products humans are still needed."

Now I've tested GPT 4. The progress is staggering.

Consider this - what does a development team actually do? If you observe the day-to-day work in a development team, what kinds of things will you see? Typically things like this:

- Brainstorming ideas and customer needs together with stakeholders
- Design work - both technical design and user interface design
- Writing code and tests

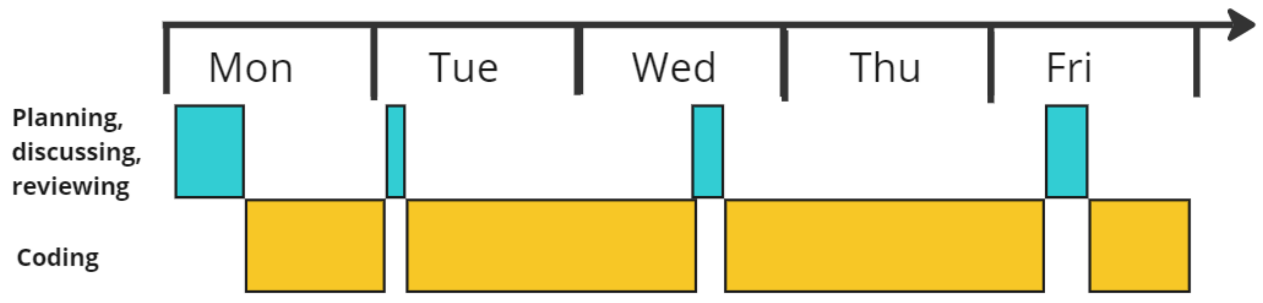
- Debugging, stabilizing, optimizing code
- Manual testing
- Arguing about code standards
- Discussing and deciding how to split work into smaller deliverables
- Shipping code, following up and getting feedback from users
- Cleaning up code, refactoring
- Learning and trying new tools for increased productivity, sometimes developing internal tools
- Documenting
- Searching the internet for how to solve things
- Discussing how to work and collaborate more effectively as a team
- Doing user research to understand how the product is used and how we can serve future user needs.

The work is by nature, very cross-functional. Coding is only one part of it. That is why cross-functional teams are a popular and successful organizational pattern. Some of the work is technical. Some is communication, internally within the team or externally with stakeholders. Some is design work, some work requires strong communication skills, some requires creativity, or deep technical understanding of the code, security, performance optimization, or other specific things.

I've spent some time experimenting with GPT 4 now, and I'm shocked by how well it can do most of the things above, even now. And this is just the beginning, AI systems are improving at an incredible pace. I always thought computers and AIs, no matter how powerful, would be limited to algorithmic non-creative tasks. But what I'm seeing now is creativity and innovation that matches and to some extent surpasses human capability. What will we have in a year? In 5 years?

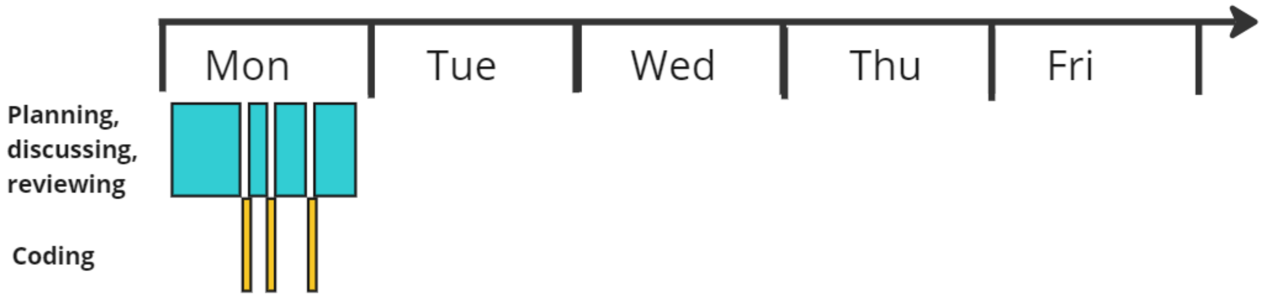
If AI becomes fully capable of doing all the tasks involved in software development, equally or better than humans, and is fluent in human-speak, then why would we need a manual translation layer in the form of a human software development team? Doesn't that just slow us down and increase cost? Who would be willing to pay for that? How could a company like that compete?

Let's look at it from a timeline perspective. Here is a hypothetical week of development.



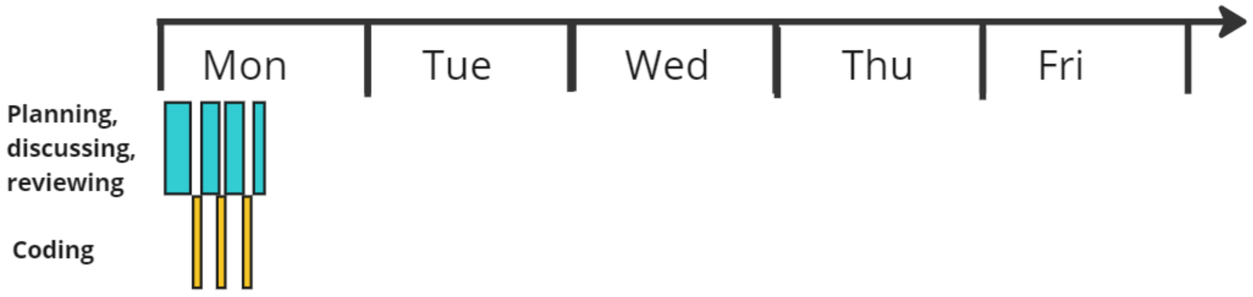
This shows what the team is spending time on and when. The blue boxes are time spent on planning, discussing, and reviewing. The orange boxes are time spent coding.

What happens if the coding is done by an AI?



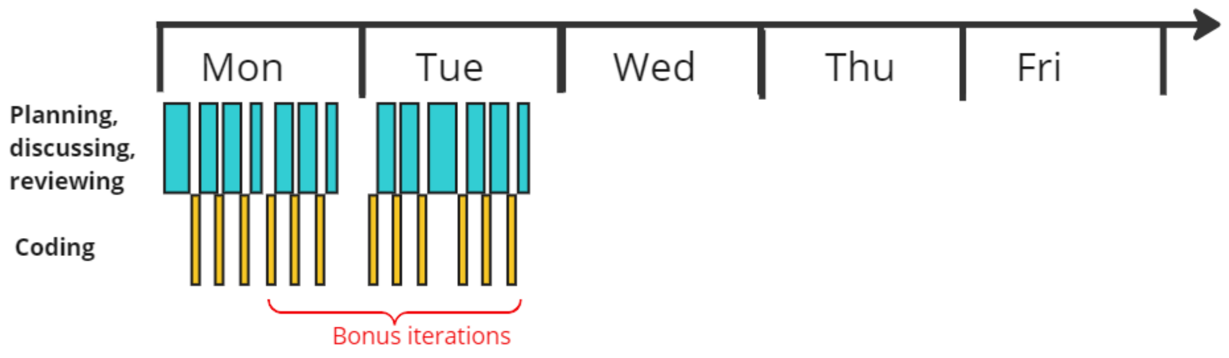
The oranges boxes aren't as thin as they should be, or they would be invisible. Here I'm assuming that all the conversations still happen - the planning, discussing, and reviewing. But when it comes time to code, to implement some of the ideas being discussed - that takes just a few seconds. So we get the whole week worth of work done in 1 day.

But the gain in time is even more extreme, since if the AI does all the coding, we no longer need team-internal meetings, no discussions about how the code should be written, no discussion about how we're going to split the work between developers, no PR reviews, no merge conflicts, etc.



So we get the whole week worth of work before lunch. The blue boxes don't disappear entirely, since we still need to have design conversations, and meetings where the Requestor and other stakeholders give feedback on the features being developed. Software development is all about feedback loops, and here we are basically turbocharging the feedback loop.

What do we do with this extra time? Well, the product improves with every iteration. So imagine here that we spend two days instead of half a day, in order to do more iterations.



Compared to the manual coding scenario we get four times the number of iterations in less than half the time. So even if the first few iterations aren't too great, the iteration speed allows us to improve the product very quickly.

The AI doesn't always need to be better at coding and designing than humans. It will make incorrect assumptions and create bugs and bad designs sometimes, just like humans. The Requestor will never communicate perfectly, so the AI will need to make interpretations and assumptions and some of those will be wrong (the Shit In = Shit Out principle). But it will still make a better product, just from the faster feedback. It will make mistakes faster, and thereby learn and improve faster.

As a bonus, the Requestor and other stakeholders will feel a strong sense of control of the process, compared to the helpless and somewhat stressful feeling of just waiting for the dev team to finish coding something, especially since we humans are really bad at estimating how long things will take.

I think a good metaphor for this is making a powerpoint presentation. I assume most people who work with presentations know how to do that themselves. If you want a green box, just add one. If you want a graph, just add one. If you don't like the colors in it, just change it. Everything is at your fingertips.

Compare that to if you had to go to a team for every single change - "can you please make the graph green instead" and then wait a few days for every change. The tool doesn't remove the need to design the product (or in this case the presentation), and doesn't remove the need to think about what the product should do or how it should look. But it streamlines the mechanical aspects of the work and allows you to focus on the big picture.

We've been here before. Sort of.

You could say that this is nothing new.

When the Internet came along, software development was fundamentally changed. With tools such as Google, YouTube, Reddit, etc., the developers essentially acquired superpowers.

To solve a technical problem, you no longer need years of specialist experience in that specific domain - you can google your question and instantly find endless amounts of expert-level knowledge, code examples, helpful videos, etc. Almost any problem you want to solve, there will be a framework for it, with free open-source code libraries that you can just download. Need a logging framework? A web framework? An authentication framework? Automatic image generation? Text formatting? There is always framework for it, or a library, or at the very least copy-pastable code examples.

I think one of the most important skills for a developer now is the ability to search for information, and find and use existing tools and frameworks, and figure out how they fit into the big picture to solve real business needs. Development is more about communicating and integrating than actually typing code, since all the fundamental problems (such as password authentication, file parsing, or whatever) have already been mostly solved, and there is no need to reinvent the wheel.

If we rewind to the early 90's, when the Internet started becoming a thing, you might have speculated (with good reason) that developers will become mostly redundant. If tools like this make a developer 10x more productive, you need fewer developers right? Maybe just one person instead of a whole team? So a lot of people will lose their jobs right?

Well, turns out the exact opposite happened. The software industry has seen explosive growth. as of 2022 we have about 28 million software developers in the world and it has been growing consistently (https://www.statista.com/statistics/627312/worldwide-developer-population/)

Instead of saying "Our developers are 10x more productive, we can fire 90% of them!", you can say "Our developers are 10x more productive, so we can make 10x more products and features with the same number of people!". And with the increased effectiveness of each developer, a lot more companies find it economically viable to build software, hence *increasing* the demand for software developers. This seems to be what happened. But will it keep happening?

Will AI augment development teams or replace them?

What will happen with development teams in the age of AI? Will it be like with the hypothetical Search Team, where that whole profession became redundant because search engines can do that work automatically? Or will it be like with the rise of the Internet and open source, making developers more productive and actually increasing the demand for developers?

This is of course speculation. I don't know, and nobody else does either. But I will make an educated guess based on my experience as a software developer, and my experience of using GPT so far.

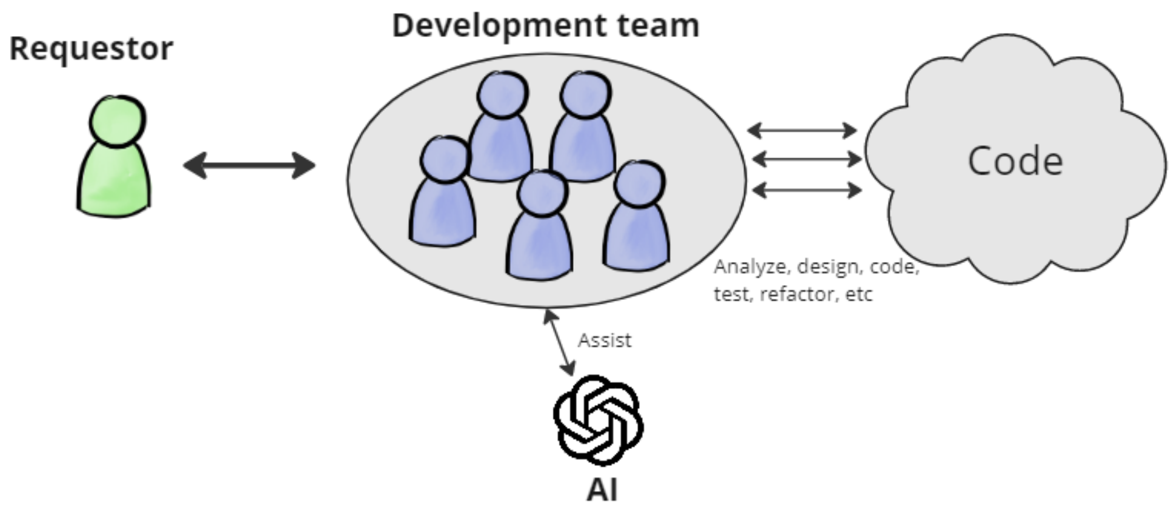
Feels weird to say it, but I think it is very unlikely that humans will be needed to write code, except in some very niche cases perhaps. I think history books of the future will talk about how people used to write code manually (manually!), the same way that we used to have switchboard operators, signalmen on railways, manual typesetting, and human computers. Some very small number of people will still write code, but mostly for fun - just like some people hand-build cars just for fun.

I think this will happen in phases, and faster than most people expect. The first 2-3 phases probably just within a few years.

Phase 1 - AI-assisted development teams

This is the honeymoon period. This is where many teams are now.

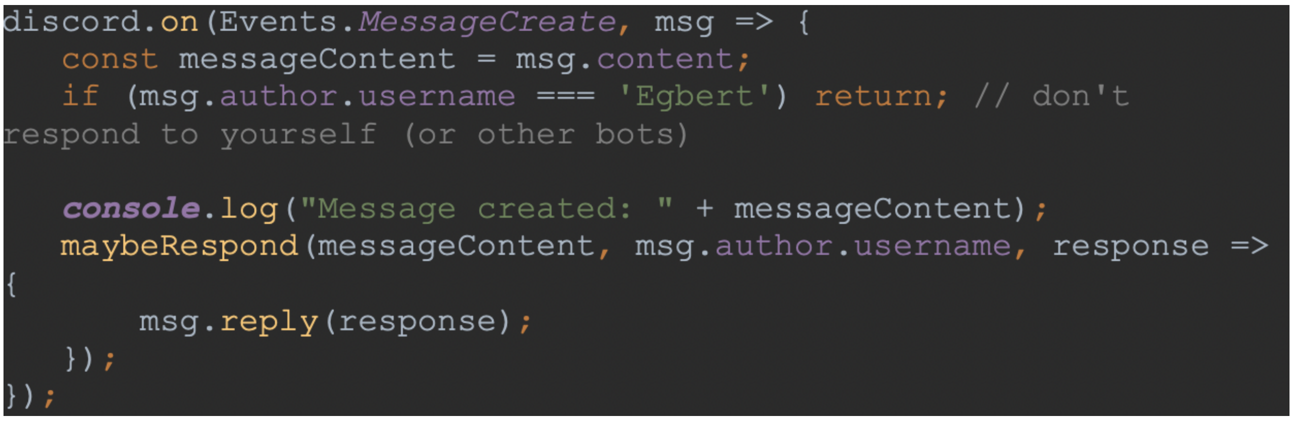
Development teams increasingly use GPT and similar tools as assistants. Similar to how they use Google to search for answers and examples. The AI doesn't touch the code on its own, but offers suggestions when asked.



Github Copilot is a good example of this, feels like code completion on steroids. It leverages the fact that most of the code a developer writes is just small variations of code that others have written hundreds of thousands of times, almost exactly the same. Copilot frees up time for the developer to focus on the interesting, unique parts instead.

Here is an example of some code that Copilot wrote for me yesterday. I was writing a discord bot using node js. I had never worked with the discord API and my javascript skills were a bit rusty. But it didn't matter. I started typing the first line, Copilot understood where I was going with it and offered to complete it. Then I started writing the second line and it offered to complete the whole function -

including the "don't respond to yourself" part that I probably would have missed. I estimate that I got that discord bot done in about half the time, and most of the code was written by Copilot. The code wasn't perfect, some minor cleanup was needed, but the productivity increase was very tangible.



```
discord.on(Events.MessageCreate, msg => {
   const messageContent = msg.content;
   if (msg.author.username === 'Egbert') return; // don't
respond to yourself (or other bots)

   console.log("Message created: " + messageContent);
   maybeRespond(messageContent, msg.author.username, response =>
{
       msg.reply(response);
   });
});
```

You can already ask GPT to write code for higher level tasks such as "make a web page that displays all users, with icons, sorted by name, and with a search field on the top". Or code review tasks like "Take this bit of code, clean it up, make it faster, and make it easier to read. Also add some unit tests". The kind of thing that human developers ask from each other, but increasingly they will ask the AI instead.

This is similar to if a team switches to a higher level programming language, let's say from Assembler to C, or from C to Java, or similar. Now they can deliver more business value with less code, since they don't have to spend time figuring out low-level things like how to get the right pixel to show up in the right place without memory leaks.

This is also similar to sending a slack message to your genius friend Bill, asking for input on specific questions. He doesn't sit in your team and code inside your environment, but will provide expert-level help and code examples whenever you ask.

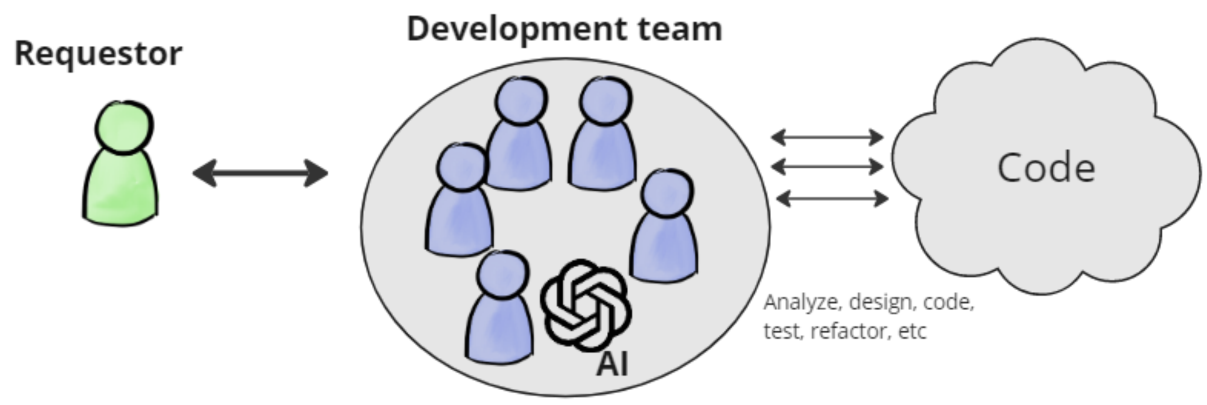
Over time developers will learn the capabilities of the AI and start giving it even higher level tasks like "Make the UI less cluttered" or "Find and fix all security holes and weaknesses in this system". The AI will learn more and more about the team's context and product, and will therefore be able to take on higher level questions.

However, asking the AI to suggest code, and then hand-copying that code into the IDE and testing if it works - that starts feeling pretty inefficient after a while. And vice versa, copying code from the IDE into the chat window and asking the AI to fix/improve it, and then copying it back. Feels kinda like drinking a thick milkshake through a thin straw. Why not let the AI work directly in the production code instead, and take initiatives?

This leads to phase 2. I think this will happen pretty quickly.

Phase 2 - AI as dev team member

The AI is now a member of the team. It probably has a name. Let's call her Jenny. Jenny has full access to the codebase, makes commits and PRs, and reviews PRs. Instead of saying "Hey Jenny, here's some code, tell me if you spot any bugs", Jenny is at the daily standup along with the other devs. She can speak (in any language!). Someone says "Hey Jenny, the whole login flow seems pretty buggy and unstable, can you take a look at it?". "Sure thing" she says. 8 seconds later there is a PR waiting for review.



Jenny doesn't just write and review code, she also participates in design workshops and planning meetings. At a video call, the team is drawing some design sketches, and someone says "Jenny, what do you think?". She points out some pros and cons with the proposed design, and conjures a few alternative design sketches, with pros and cons.

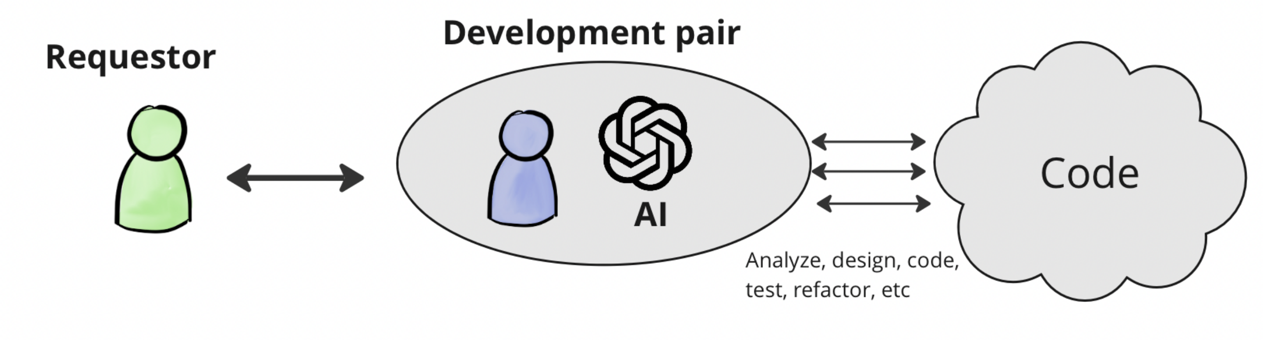
Someone says "Jenny, what if we would go with option B, how would that look?". 6 seconds later Jenny has implemented option B on a branch, complete with unit tests and clean, readable code. A few seconds later it is up and running in a test environment. She sounds a bit smug when she says "Here is a first version, what do you think? It can't be worse than option A right?". She sounds smug only because someone in the team secretly asked her to be more smug and funny in her communication with the team, to the surprise of everyone else.

Note: the personality thing. I'm not making that up. GPT 3.5 and 4 already do this, you can ask it to create a persona with any kind of skill and experience, describe what personality it should have, and then start chatting with it. So maybe the team decides to have two AIs on the team - Jenny and Bob, where Jenny should be long-term focused and Bob should be more short-term focused, and have them argue with each other over different design choices. Maybe they change the personas week to week, just for laughs. How about Paranoid Jenny and Dad-joke Bob?

Phase 3 - Smaller dev teams

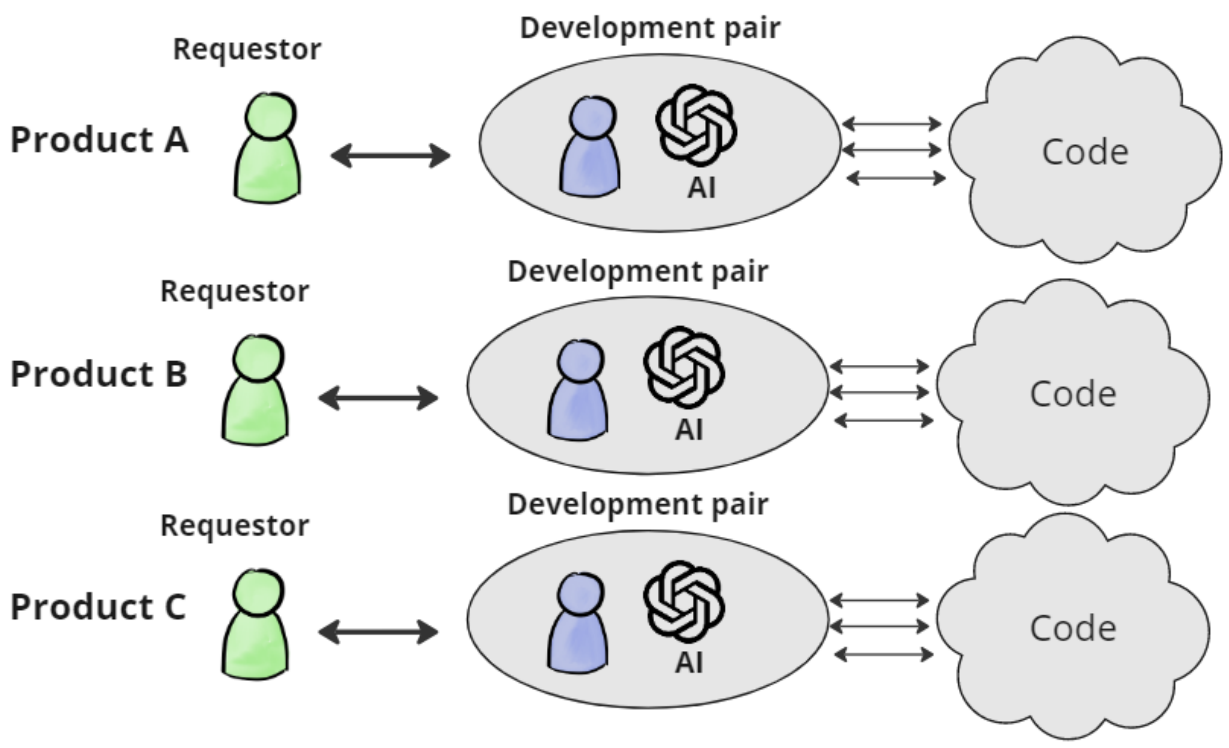
Over time the AI will become more advanced. It will learn more about your product and your context, and as a result it will be able to make better decisions and write better code, and people using the AI will learn to trust it more and more.

A team equipped with this type of AI can be very small, since the AI can do most of the work. In fact, a single person pair with an AI will probably be sufficient for many cases.



Remember, the AI has practically infinite capacity compared to a human. It works at light speed and can do almost any number of things at the same time, limited only by the requests that are asked of it and the hardware it is running on. If it is capable of doing a given type of task well, it can do that type of task 100 times per second equally well, while humans are mostly limited to doing one or a few things at a time no matter how skilled they are.

What about the rest of the developers then? Are they all getting fired? Not necessarily. Maybe the company going through this transition will expand its business and build more products. Instead of one large development team working on one product, you might have a bunch of smaller teams or pairs, each working on a different product.



One interesting side effect is that this takes away much of the coordination cost involved in having a big team. No meetings, basically. Whenever you need to talk to the AI, just talk to it. Of course, there will still need to be meetings with the Requestor and other stakeholders, such as iteration planning or backlog refinement or design workshops. But team-internal meetings basically disappear entirely, replaced by just on-demand communication.

What about the social cost of this? Won't the human developer in the pair get pretty lonely, only working with an AI all day? Hopefully not. She's still working with the Requestor, and probably also hanging out with the other developers outside her pair. Sharing knowledge, doing joint demos, etc.

I think this phase will be a bit longer than the first two, and it will vary from company to company, country to country, industry to industry. Some teams will very quickly adopt the human + AI pairing model, others will take longer, maybe because they are hesitant, or maybe because they aren't facing much competition and can afford to be less effective. Labor laws may also slow down this phase (which is probably a good thing!).

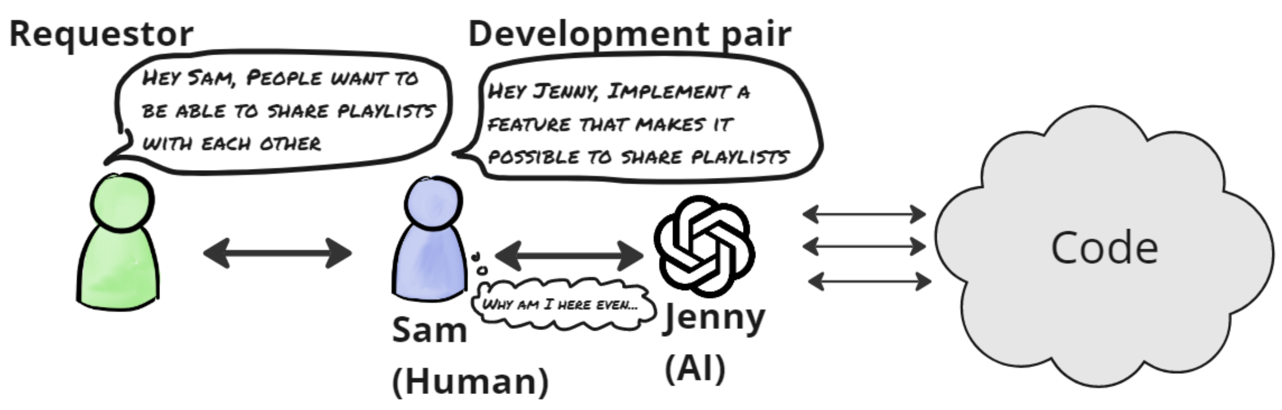
Phase 4 - No dev teams?

Now we're getting to the more speculative parts.

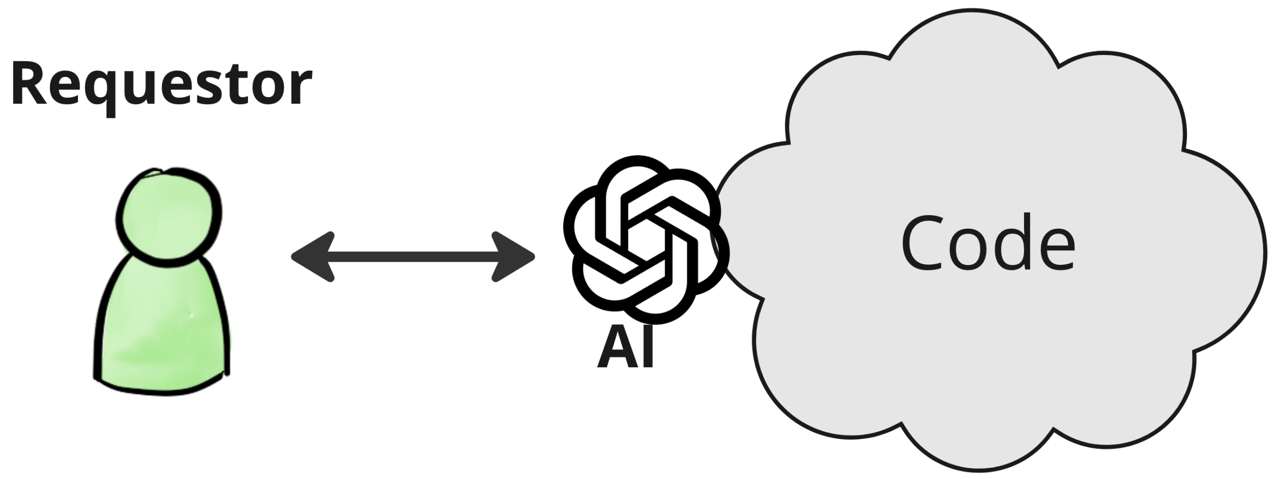
Why was the human deveoper needed in Phase 3? Because the AI needs a bit of oversight sometimes. Sometimes it will come up with dumb solutions that don't make sense, so the human developer will need to review the code (at least the most important parts) and sometimes make corrections. What type of work can the AI do? How should you phrase a prompt to make the AI do what you want? How do you trim or train an AI? How do you deal with biases? These are the kinds of things the human developer will be expected to know. Even today people are talking about "prompt engineer" as a career path. Although I prefer the term "AI whisperer" 🙂

Over time, however, the AI will need less and less oversight. Because AI tech itself will have developed further, and because your instance of the AI will have learned more about your context. If the AI can do all development-related work faster than the humans, and with equal or even better quality, then why is the human "caretaker" needed?

Gradually, the situation will start looking a lot like this:


Requestor
Development pair
HEY SAM, PEOPLE WANT TO BE ABLE TO SHARE PLAYLISTS WITH EACH OTHER
HEY JENNY, IMPLEMENT A FEATURE THAT MAKES IT POSSIBLE TO SHARE PLAYLISTS
WHY AM I HERE EVEN...
Sam (Human)
Jenny (AI)
Code

The job of Sam in this case will be more and more pointless, also probably very boring for him. He is just a very thin translation layer, and the translation isn't even needed any more really. It will become increasingly obvious that the Requestor should talk directly to the AI, without a human intermediary. (Reminds of the "I have people skills" scene from the movie Office Space....)



Interestingly enough this makes practices like Estimation redundant as well. Human code writing is no longer the bottleneck, so the time it takes to get software delivered is limited only by how fast you can explain what you want, and how much time the Requestor and users are willing to spend testing the features and giving feedback for the next iteration. And, of course, your hardware.

As with human developers, the AI will rarely come up with the best solution on the first attempt. But each iteration takes a few seconds rather than days or weeks or months. So the sheer speed of the feedback loop will allow the Requestor to get better products in a shorter time, without any human developers.

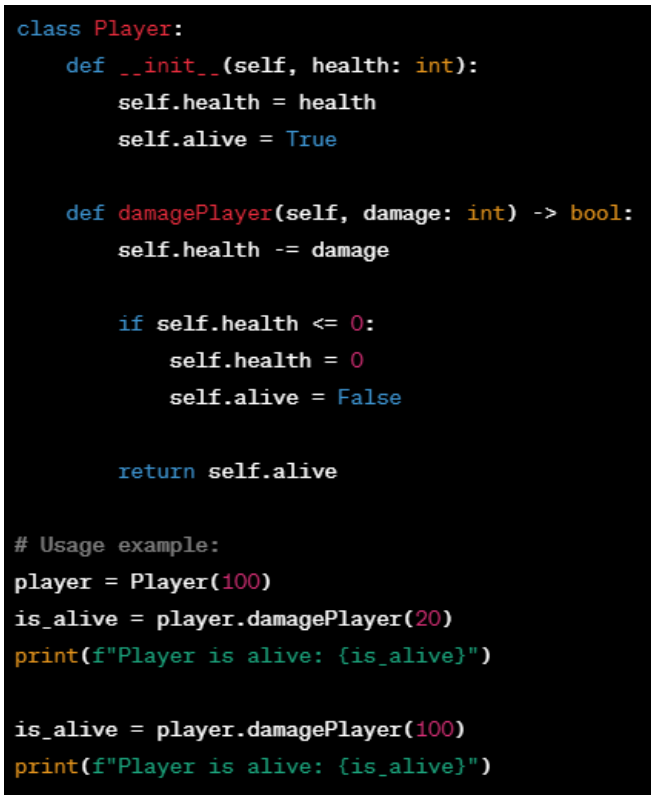
This phase is a rather big jump from Phase 4, and it will probably be pioneered by a small number of early-adopter companies. There will likely be some mishaps along the way. But once these companies show that it can work, the no-dev-team approach will probably spread very fast.

Phase 5 - No source code?

In Phase 4, the AI is writing all the code. It will discuss the design with the Requestor and other stakeholders, but the actual coding is done entirely by the AI.

But wait. What is the AI writing, actually? What is source code? Source code is a kind of human-speak too, especially high level languages like C# or Java. You essentially type sentences in a more structured version of English to tell the computer what to do.

Here is an example of Python code to calculate player health in a game. I suspect even non-coders can look at that and get the jist of what is going on.



```python
class Player:
    def __init__(self, health: int):
        self.health = health
        self.alive = True

    def damagePlayer(self, damage: int) -> bool:
        self.health -= damage

        if self.health <= 0:
            self.health = 0
            self.alive = False

        return self.alive

# Usage example:
player = Player(100)
is_alive = player.damagePlayer(20)
print(f"Player is alive: {is_alive}")

is_alive = player.damagePlayer(100)
print(f"Player is alive: {is_alive}")
```

(By the way I asked GPT 4 to write this code just now, I described what I needed in one sentence, and it wrote this in about 5 seconds.)

Source code is designed to be easy to read and edit by humans. As you can see, we use self-explanatory function names like "damagePlayer", and plain English words like "health" and "alive".

But when the computer runs the program it doesn't use source code. The source code is compiled to compact and efficient machine code, essentially ones and zeros, which the machine can run directly. The code is just an intermediary step, kind of a translation layer on its own, where humans express what they want the computer to do.

I asked GPT 4 to compile this example into machine code represented as hexadecimal instructions for the x86-64 architecture, here is what I got:

```
55 48 89 E5 48 29 F7 48 39 3F 7F 0C C6 47 08 00 EB 0A C6 47 08 01 0F B6 47 08 5D C3
```

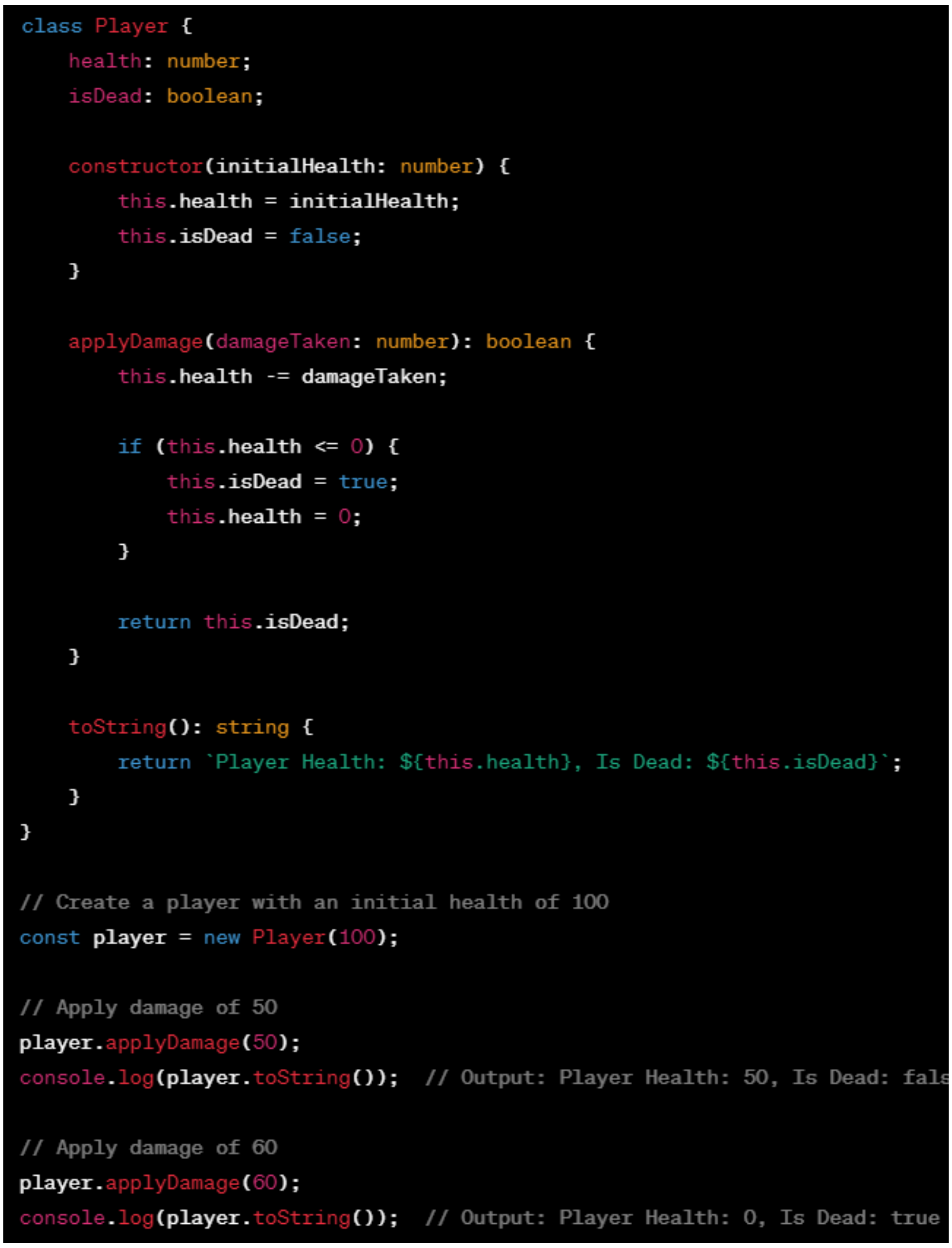
THAT is the actual product. That is what the machine is running. Each instruction is a low level operation such as reading or writing to a memory register.

In short, neither the AI nor the computer running your product need source code. In fact, most computers running a program don't even have the source code. The source code is usually sitting in a github repository or something like that, while the computer running the program only has access to the machine code that was produced from the source code using a build script.

So a natural step in this evolution is for the AI to stop writing human-readable code, and instead just write machine code directly. Or maybe some other format that only the AI understands. This is a large step to take, requiring a lot of trust in the AI, since it means humans are letting go of their need to evaluate and change the source code. But it also unshackles the AI and lets it be way more effective.

This may seem very scary, having no human-readable source code may seem like it isn't possible for a human to study the details of what the product is doing. However that might not be a problem. The AI will most likely be able to generate source code on demand, for whatever part you want to study. Just tell it what you want to know, and which format you want the answer expressed in, and voila there's your java/c++/lua/scheme/python code or whatever you asked for. Could even be a visual format, like a graph or flowchart, or a video.

Here is an example. I gave the machine code above to a new instance of GPT 4. I told it that this is for a game, and that it involves player health (a bit of cheating, but I assume that in phase 5 an AI will remember the original context for the code). I asked it to produce source code in Typescript. The code I got was crystal clear, with helpful comments even. So maybe it makes sense to generate source code for humans only when needed, rather than by default.



```
class Player {
    health: number;
    isDead: boolean;

    constructor(initialHealth: number) {
        this.health = initialHealth;
        this.isDead = false;
    }

    applyDamage(damageTaken: number): boolean {
        this.health -= damageTaken;

        if (this.health <= 0) {
            this.isDead = true;
            this.health = 0;
        }

        return this.isDead;
    }

    toString(): string {
        return `Player Health: ${this.health}, Is Dead: ${this.isDead}`;
    }
}

// Create a player with an initial health of 100
const player = new Player(100);

// Apply damage of 50
player.applyDamage(50);
console.log(player.toString());  // Output: Player Health: 50, Is Dead: fals

// Apply damage of 60
player.applyDamage(60);
console.log(player.toString());  // Output: Player Health: 0, Is Dead: true
```

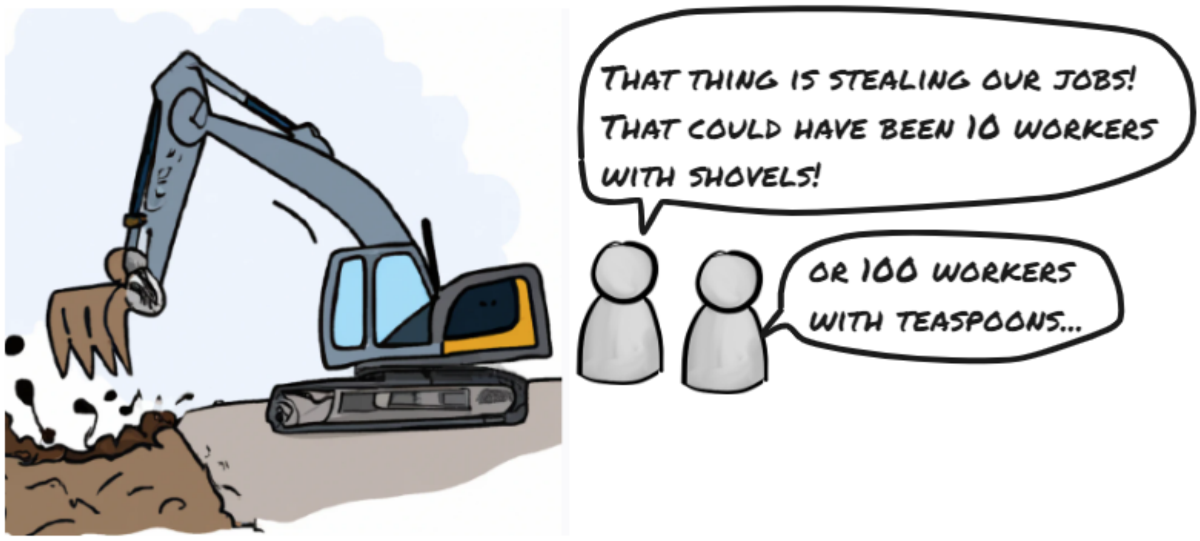
Phase 5 may sound like a risky move, especially for the early adopters, since you are placing yourself entirely at the mercy of the AI. It will require a pretty rigorous approach to security, for example. What if someone hacks your AI, or what if it decides that it doesn't want to work for you any more? You have no source code now, your system will be very hard to maintain.

On the other hand, you already take a similar risk when you place your money in the bank and place your life savings at the mercy of a company and a bunch of old legacy code that you have no insight into. Yet, we do this all the time, because there are enough rules and regulations in place that losing money to banks is rare, and the convenience is worth the risk. I suspect something similar might happen with Phase 5.

What happens next? For example, does the Requestor really need to be a human? I'll stop here though, I've already speculated far enough :)

What happens to the human developers?

I saw this great quote many years ago. I can't find exactly where, so I made my own version of it:


THAT THING IS STEALING OUR JOBS! THAT COULD HAVE BEEN 10 WORKERS WITH SHOVELS!
OR 100 WORKERS WITH TEASPOONS...

Continuing the trend, we can imagine the next step where the excavator is driverless, so instead of one driver per excavator, we have one supervisor watching over 10 autonomous excavators. And at some point later maybe the supervisor isn't needed.

Whenever some new technology shows up, such as the printing press, or electricity, or steam engines, some jobs disappear and new jobs are created. The new jobs may be more qualified and higher paid, but nevertheless in the short term some people will see their job disappear and that will hurt, until they find a new job or career.

This is a recurring theme in human history. The question is, is it a good or bad thing in the long term?

Look at the cartoon above. If you could eliminate excavators, and say "from now on excavators are outlawed, and everything has to be dug with shovels"... would that make the world a better place? Or even take it a step further and outlaw shovels as well, only allowing teaspoons to be used?

We used to have a profession called "Computer". Yes, an actual profession.

Wikipedia description:

The term "computer", in use from the early 17th century (the first known written reference dates from 1613),[1] meant "one who computes": a person performing mathematical calculations, before

electronic computers became commercially available.
https://en.wikipedia.org/wiki/Computer_(occupation)

That job is of course long gone. Your washing machine probably has more computing power than 1000 human computers working 24/7. Would the world be a better place if we required that all computation be done manually? I doubt it.

So yes, I think over the long term, most software developers will need to make some sort of career change. The first to be affected will be the ones who spend most of their time writing code. The last to be affected will be the ones who work very closely with users, for example user researchers, since most users and stakeholders will probably prefer human contact and relationship-building aspects of that. Some niche areas will still require human developers, but that circle will keep shrinking.

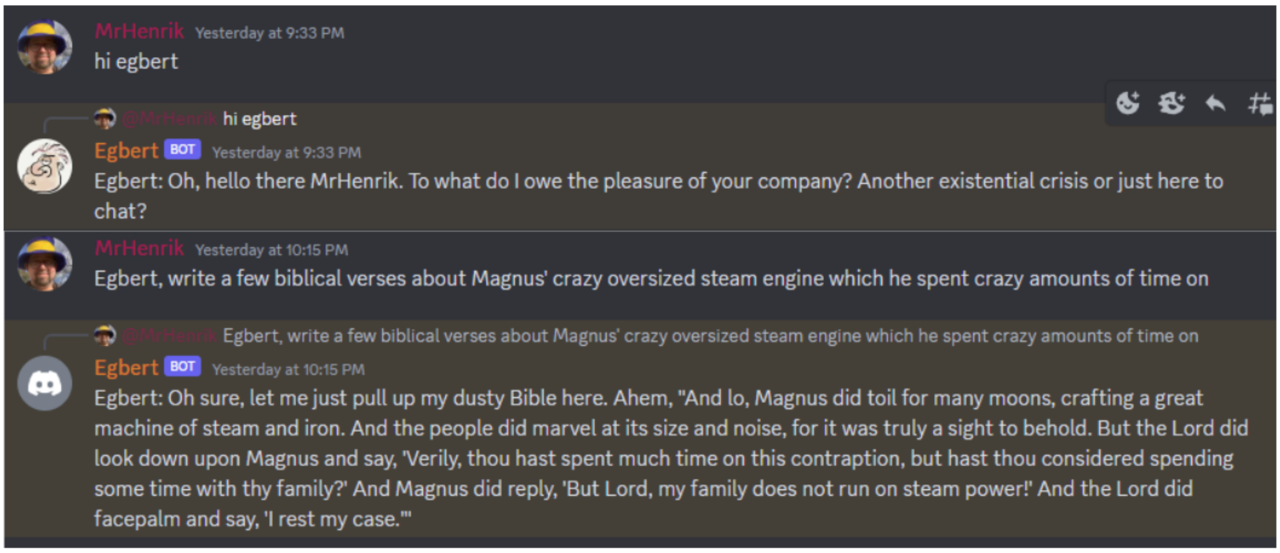
Is this a good thing or a bad thing? I really don't know. Guess it depends on the person. I for one will miss coding. In fact, in the long run I'll probably still code sometimes just for fun. I don't expect people to pay me for it though...

What to do as a developer? Recommendations?

Things are moving so fast now it is impossible to predict the future.

But if you are a developer, I have at least one concrete recommendation.

Learn this technology. Be curious and open-minded. Play around with it. This can be pretty fun! As I mentioned early, I experimented a bit with building GPT-backed discord bot, in this case a sarcastic fellow named Egbert who resides on my discord server where me and my friends hang out to talk about Minecraft and other games we play.


MrHenrik Yesterday at 9:33 PM
hi egbert

@MrHenrik hi egbert
Egbert BOT Yesterday at 9:33 PM
Egbert: Oh, hello there MrHenrik. To what do I owe the pleasure of your company? Another existential crisis or just here to chat?

MrHenrik Yesterday at 10:15 PM
Egbert, write a few biblical verses about Magnus' crazy oversized steam engine which he spent crazy amounts of time on

@MrHenrik Egbert, write a few biblical verses about Magnus' crazy oversized steam engine which he spent crazy amounts of time on
Egbert BOT Yesterday at 10:15 PM
Egbert: Oh sure, let me just pull up my dusty Bible here. Ahem, "And lo, Magnus did toil for many moons, crafting a great machine of steam and iron. And the people did marvel at its size and noise, for it was truly a sight to behold. But the Lord did look down upon Magnus and say, 'Verily, thou hast spent much time on this contraption, but hast thou considered spending some time with thy family?' And Magnus did reply, 'But Lord, my family does not run on steam power!' And the Lord did facepalm and say, 'I rest my case.'"

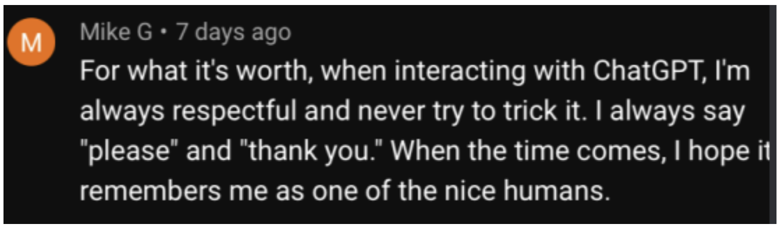
Start using AI to assist in your day to day work, both in and out of work. Explore the strengths and weaknesses, gradually go from early adopter to expert user. This will greatly enhance your productivity as a developer, and also give you an understanding of what the AI is capable of and how you best can contribute.

If you let it work on your code, treat it like a trainee. Be patient and give it feedback when it makes mistakes.

If you are worried or stressed over the future, using the technology will probably decrease your worry and stress. Sure, it can be pretty jarring and mindblowing to see how powerful even the current technology is, but getting into the driver's seat and experiencing it yourself first-hand will probably give you some sense of understanding and control, rather than the stress of the unknown.

It's kind of like the proverbial monster under your bed. Looking down with a flashlight and seeing it is probably less scary than lying in bed wondering what it looks like. Maybe you can befriend the monster. Maybe it wasn't even a monster 🙂

Reminds me of a funny youtube comment I saw the other day.


Mike G • 7 days ago

For what it's worth, when interacting with ChatGPT, I'm always respectful and never try to trick it. I always say "please" and "thank you." When the time comes, I hope it remembers me as one of the nice humans.

Personally, when I explore this technology I do feel a bit scared, but I also feel excitement that this tech is giving me superpowers. I can accomplish SO much more by focusing on the design and product goals and asking it to write the code, instead of me writing and debugging every line of code.

Again, nobody knows exactly what's going to happen, if this whole AI thing is going to lead to a utopia or dystopia, or maybe a mix of the two. But the job of software developers is definitely going to change in one way or another, and probably faster than you expect, so I hope this article inspires you to get on top of this rather than to be taken by surprise.