WIRED Consulting.

The rise of the codebots:
How AI is changing software development

thoughtworks

Ethan Mollick is not a professional coder and nor does he pretend to be one.

Yet recently he was able to create a program for an elementary but playable video game in honour of *Doom*'s 30th anniversary.

How? Simply by prompting a chatbot.

The interaction was no more complicated than this:

chatgpt, make me an interactive game that is a homage to *Doom*, it should be complete and playable, but within what you can actually do. give me a zip file with html, css, and javascript (or take another approach). You can be incredibly creative and are very curious and great.

[AI explains how to do it]

no you need to do it. don't be lazy. make me a working game. you can create downloadable files. and you can do all of the things you said.

[AI creates zip file]

make it more *Doom* like.

[AI creates zip file]

that works! but can you make it more *DOOM* like and more cool in design?

[AI creates zip file] …

A

rtificial Intelligence just lets you code by intent" says Mollick, an Associate Professor of management at the Wharton School of the University of Pennsylvania, and the author of Co-Intelligence: Living And Working With AI.

The chatbot Mollick used was ChatGPT, which is underpinned by generative AI. This is an emerging, powerful class of algorithms that can create new content—text, music, video, imagery—and as Mollick's experience shows, it has an aptitude for software development. 'Large language models' (LLMs), a category of deep learning model that powers many gen AI tools, have an astonishing ability to write and complete computer code.

Thanks to a new range of generative AI coding assistants, the technology is already making programming faster and more productive. "AI represents a significant leap forward for software engineering," says Mike Mason, Chief AI Officer at global technology consultancy Thoughtworks. "The rapid adoption and constant advancement of AI tools for developers shows the transformative impact this is already having, and will continue to have, on our industry."

But what precisely will those changes look like and how far will they go? Will there be trade offs that need mitigating? And what will all this ultimately mean for the role of developers and the kind of software they create?

After all, some believe generative AI represents a technological inflection point as significant as the shift from command line to graphical interface or even the birth of the web. Perhaps, they suggest, it will come to change computing as we know it.

But first, a look at the here and now...

"AI REPRESENTS A SIGNIFICANT LEAP FORWARD FOR SOFTWARE ENGINEERING"

Mike Mason, Chief AI Officer at Global Technology Consultancy Thoughtworks


1
10
20
30
40
42%
100%
20%
100
10
2000
1000
900
800

Hello, (brave new) world!

There is a reason generative AI has proven inherently well-suited to coding. Writing software is a language-driven task, but one that is highly structured and rule-based, and there are massive pre-existing repositories of viable code for the models to draw on.

The last two years have thus seen the rise of specialized tools that can complete code from partial inputs in real time or even script it from scratch. They are smart enough to understand context, so can write code that aligns with developer requirements, and often include a chat interface for asking coding-related questions.

"Like autocomplete on steroids", is how Mason characterizes AI coding assistants. They take a lot of the drudgery out of programming, speeding up routine coding processes, detecting errors and providing an on-the-spot resource for troubleshooting and problem-solving. "There are times when it literally feels like it has read my mind regarding the block of code I wanted to write," says Mason, who started coding at the age of six.

One crucial advantage of these tools is that they can eliminate the need to leave the creative bubble of the Integrated Development Environment (IDE), the programming suite used by coders, in order to trawl search engines or other sites to find solutions to problems. Leaving the IDE can break a programmer's all-important 'flow'—a state of full immersion in the task at hand, associated with higher performance.

And they are proving hugely popular. As of January 2024, one of the leading tools now has 1.3 million paying subscribers around the world—a 30 percent increase quarter on quarter—and counts more than 50,000 companies as customers. According to a recent survey, 95 percent of developers are already using AI tools to write code.

Increasingly, coders see them less as optional extras than as necessities. "It wasn't like our developers were asking me for it, they were begging me!" says Andreas Nauerz, CTO and Executive Vice President at Bosch Digital. His division implements digital solutions across the entire Bosch engineering and technology group. After evaluating the potential risks and benefits, the company onboarded a leading AI coding tool in December last year and is using it as a programming aid across many of its projects. "It's a huge

"IT'S A HUGE EFFICIENCY BOOSTER"

Andreas Nauerz, CTO and Executive Vice President at Bosch Digital

efficiency booster," he says. "And it makes people happier, because they can focus on actual problems, and not just the everyday boring tasks that you have to do."

Evidence suggests code-completion tools are best seen as providers of useful suggestions for further development, rather than as writers of finished code, but nevertheless the benefits are quantifiable. A study by a management consultancy found that 75 percent of programmers who had adopted AI coding assistants said the tools had met or exceeded their expectations, with 'speed to market' seen as the biggest bonus. In a survey of more than 2,000 developers by a leading vendor, 88 percent responded that they were now more productive when using an AI coding assistant. A comprehensive paper about measuring the impact of code-completing AI on productivity concluded that, while "developer productivity is a multidimensional topic that cannot be summarized by a single metric" the reported benefits of AI coding suggestions "span the full range of typically investigated aspects of productivity, such as task time, product quality, cognitive load, enjoyment, and learning," noting that "perceived productivity gains are reflected in objective measurements of developer activity."

When AI isn't so intelligent

Being a good software engineer is, of course, about much more than speed—and here's where the picture gets more complicated. A recent investigation by a code analysis company found that the general quality of code has decreased since the introduction of AI assistants. Indeed, a benchmark study by another code analysis company looked at refactoring—the process of improving the design of existing code without changing its behavior. It found that existing AI solutions only deliver functionally correct refactorings in 37 percent of cases.

"One of the trade-offs to the usefulness of coding assistants is their unreliability," says Birgitta Böckeler, Global Lead for AI-First Software Delivery at Thoughtworks. "The underlying models they use are quite generic and based on a huge amount of training data that is not always relevant to the task in hand. And large language models also make things up—they 'hallucinate'."

What's concerning is that the mistakes AI makes can


STL

be subtle and therefore hard to spot. While the code it produces is best seen as merely a starting point, it can look correct and convincing even when it contains errors, so if a developer doesn't have their wits about them, they can let bugs slip past unnoticed. This cuts to the paradox of these tools. While they make coding more accessible to the less experienced, it is precisely the less experienced who are more likely to fall victim to their shortcomings.

"I worry about giving coding assistants to inexperienced developers," says Rebecca Parsons, CTO Emerita at Thoughtworks. "Someone who's been coding as long as I have can spot off-by-one errors and little things like that, but that's not so easy if you're not experienced. These coding models really don't know what good code is, so there's a danger that we end up getting a lot of poor quality code with more bugs."

The security concerns are obvious. Academic research has found that coders using AI assistants not only write significantly less secure code than those not using them, they are simultaneously more likely to believe they are writing secure code. AI can hallucinate fixes for common vulnerabilities and leave off essential validation checks, exposing systems to hackers. And as younger coders come to depend on these tools more and more, this might also be storing up a more fundamental security problem for years to come. "I see a huge risk of a degradation of coding skills for the future," says Nauerz. "The more people lose the deep knowledge of how systems work, the more we lose control of them and the more we are exposed to new attack vectors."

There is an ethical-cum-legal consideration, too. Examples of AI being trained on copyright-protected data have been making headlines in recent months, and coding assistants now also find themselves in the dock. One of the leading vendors has been targeted in a proposed class action lawsuit that describes the vendor's actions as "software piracy on an unprecedented scale". This refers to the long lines of licensed code that can be regurgitated by assistants without credit. However, the case is in its very early stages.

Clearly, as developers venture further into this brave new world, it will be imperative to mitigate the risks. Companies will need to have clear policies around the use

of AI—keeping humans in the loop, not using models that incorporate prompt data—and developers should adopt a mindset of healthy skepticism when it comes to reviewing AI output.

When Böckeler began working with an AI assistant, she decided to give it a persona, as if it were a new colleague. It helped crystallize how she should approach working with it. She defined it as eager to help, stubborn, very well-read but inexperienced, and unwilling to admit when it doesn't know something. She then put the description, together with different animal characters, into AI image generator Midjourney, which produced a picture of a grinning, slightly manic donkey surrounded by books.

Where next? From scripts to systems

Developers do more than writing code. On average, programming occupies less than a third of their time. The rest is spent on a variety of tasks including testing, debugging, talking to end users, designing new solutions, or learning additional skills. And then there are the bigger-picture creative tasks: what does the software actually need to achieve in the first place, and is the client thinking about that in the right way?

"AI does a great job in terms of coding assistants and it's currently on a path of increasingly providing more relevant suggestions," says Amjad Masad, founder and CEO of AI-assisted coding platform Replit. "But in terms of what features you want to build, the structure of the product, integration tests and components, all that stuff, AI currently is still at a very early stage."

In the future, this is likely to become a growth area. Thoughtworks has recently been running a number of experiments across the development lifecycle to see where generative AI might make a significant difference. Early results have shown a 10- to 30-percent increase in productivity—and not just in coding. In China, it has used generative AI as a knowledge-sharing mechanism, allowing all the context of a project to be synthesized for intelligibility. The company has also used generative AI as an ideation tool, constructing it in such a way that it can provide ideas on a spectrum ranging from conservative through pragmatic to blue-sky.

"WE HAVE TO DO SOME MAJOR REINVENTION OF HOW WE THINK ABOUT SYSTEMS"

Ethan Mollick, Associate Professor at the Wharton School of the University of Pennsylvania

Bosch Digital is also currently exploring that question, too. Nauerz believes AI will not only enhance the full gamut of existing processes, but also underpin the processes we'll require tomorrow, such as bias detection. "It's a challenge that's specific to AI," he says, "and we'll need a new tool for it."

As generative AI rolls through software development, there will be a number of implications for IT departments and ultimately the C-suite. Most immediately: what happens to all the time that is saved? Perhaps this will be reallocated to those creative tasks, perhaps it will be spent on solving associated challenges around security, or perhaps it will simply let teams do more, faster. All organizations will have different priorities, but clearly it will be important to be intentional.

Management will also have to resist seeing this as a panacea for productivity. Faster coding can, counterintuitively, create bottlenecks in the pipeline. "Most organizations actually have a fairly slow test and deploy cycle to verify that the software is correct and actually put it into production," says Mason. "So if you've got a slow test and deploy cycle, but you're rapidly spewing out code, you're actually going to make that problem worse, because you're going to have more things to put out in each release, and you're going to have bigger releases and more risk."

And then there are larger, organizational questions. Software development currently tends to rely on Agile—a term that refers to both a set of values and principles, as well as established project management and engineering methodologies. The latter come in a variety of forms but tend to emphasize rapid iteration by cross-functional teams, with continual customer evaluation. The introduction of AI may change those methodologies. "All of the methods we use for organizing work only take into account one kind of intellect, which is human intellect, but that's no longer the case," says Mollick. "So we have to do some major reinvention of how we think about systems." What this means is that Agile processes are likely to change if, say, generative AI can gather client requirements or if we can simply tell it about a product and ask what it would think about it if it were a specific kind of person. Within this paradigm, current Agile methodologies may suddenly look decidedly non-agile.

Visions of the future

Will we one day reach a stage where anybody can create software using generative AI, regardless of whether they understand a programming language? And if that's the end game, what might the path from here to there look like?

An intermediary step takes the concept of an AI coding assistant and gives it agency. 'AI agents'—chatbots that can autonomously use applications, complete tasks, and carry out multi-stage processes—are a hot topic in the field, with major players investing heavily in bringing the concept to life. Replit's Amjad Masad believes it holds enormous potential for software developers; start-ups are already building agents designed to manage engineering processes from writing code to debugging and deployment. Masad foresees a time in the relatively near future when programmers become "god-like" project managers operating on an architecture and design level who "command armies of software agents to build increasingly complex software in insane record times".

In the more distant future, perhaps agents may also be able to handle higher-order goals: rather than giving the AI a defined picture of what you want the software to do, you could tell it what function you ultimately want it to fulfil, and then everything would happen automatically from there. If a tool like that could produce even the most complex software at a production-grade level, 'programming' as it is currently understood, would ultimately cease to exist.

The big questions

This discussion has now entered the realm of the unknown, and it raises a number of profound questions. Here are three:

1. Will AI really get that good?

The possibility of large language models transforming the nature of software development to the fullest extent described depends on whether those models can continue to make giant leaps in abilities. Arguably, for tools to operate at this hypothetical level, they would need to approximate human-level intelligence. Some believe this will emerge spontaneously as LLMs have more data and more compute thrown at them. Others, however, are more skeptical. Right now, it's impossible to say one way or the other. "At some point, AI will definitely hit a limit, and when it hits that

./"WILL PROGRAMMING AND SOFTWARE AS WE KNOW IT CHANGE COMPLETELY?]

limit, the question will be: is that the end of the ballgame?" says Mollick. "Or is there a lot of extra slack that no one's bothered developing yet because everyone's rushing to build bigger models? We don't have the answers yet."

2. Will this inundate the world with poor quality software?

There is a concern that we might end up in a vicious cycle in which AI churns out bug-ridden, unsecure code, which then becomes part of the pool of code that other language models draw on. Böckeler doesn't dismiss that anxiety. "We depend so much on software, and that sometimes makes me nervous," says Böckeler. "Ideally AI will help us build better, more reliable software, but what if it just makes everything worse, because anybody in theory will be able to create code? We always say to our clients that AI amplifies indiscriminately, so you have to be careful where you apply it."

3. What does generative AI mean for the nature of software itself?

The basic idea of software has remained the same since the dawn of computing. If you want to perform a task you either buy or build a specific tool to perform that task. What's different about generative AI is that it can perform a wide range of tasks across many different data modalities—some believe potentially any task. Does that mean it can become smart enough to operate as a universal 'super-app', a single interface you use to do everything? If so, one might imagine we have less software, not more of it, and the question for developers would be what the user experience ought to look like and how best to build for it. Then again, does history suggest this is a realistic outcome? Will LLMs not be unbundled into task-specific tools just as spreadsheets were unbundled into specialist apps such as bookkeeping and CRM software? Perhaps the extent of the shift will depend on whether generative AI causes voice interfaces to take off, suggests Mike Mason. "The digital 'smart assistants' we have on our phones and around our houses today are smart in some ways but annoyingly limited in others. With advances in AI they may become our primary mechanism for interacting with the world," says Mason. "Rather than using an airline's app we

might just ask our personal AI to book flights, check us in, or find our lost luggage. This absolutely opens up a revolution in what software looks like and how we'll interact with it."

The path ahead...

Generative AI is now the technology world's center of gravity, but it hasn't occupied that position for long and it is evolving fast. It is an arena still defined by questions more than answers. It seems a safe bet to say that the role of software developers will change— and that their expertise will increasingly be deployed at a higher level of abstraction across the development process—but in what ways and to what extent? Time will tell.

Of course, the most pressing concern for those in the profession is whether, fundamentally, 'software developer' will remain a job. Rebecca Parsons is optimistic.

"The world is going to continue to change, and we're going to be faced with new challenges," she says. "But the thing we have to realize about these generative AI models is that they effectively work on what's happened in the past." So, when novel situations arise in the future and something that nobody has seen before needs to be represented in code, who are we going to call?

"The skill that we as software developers have is to imagine how to map what's in the real world and the abstractions that are needed," Parsons says. "There's still going to be a role for humans. At least for a very long time."

"THERE'S STILL GOING TO BE A ROLE FOR HUMANS. AT LEAST FOR A VERY LONG TIME"

Rebecca Parsons, Chief Technology Officer – Emerita at Thoughtworks

WIRED Consulting.

WIRED is where tomorrow is realised. It is the essential source of fresh thinking and deep expertise on the technological, scientific and societal trends that are changing our world. Consulting is a division of WIRED that brings the unique WIRED network, insights and brand to commercial organisations — helping them to build internal knowledge, develop strategy and create thought-leading content that positions them at the cutting edge.

/thoughtworks

Thoughtworks is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are over 10,500 Thoughtworkers strong across 48 offices in 19 countries. For 30 years, we've delivered extraordinary impact together with our clients by helping them solve complex business