Is Coding Dead? An Exploration of AI Programming Agents and the Future of Human Developers

Will Adkisson

Over the past three years, the rapid development of Large Language Models (LLMs) has dramatically changed the way we work. Tools like OpenAI's ChatGPT are now central to all kinds of different workflows, assisting with tasks ranging from writing to data analysis. As the rate of improvement for these models shows no sign of slowing down, many people fear for their job security. One of the most threatened careers right now is software engineering (SWE), as recent innovations in LLM training have made it possible for AI models to complete programming tasks start to finish with just a single prompt. In this paper, I seek to analyze the proficiency of today's state-of-the-art (SOTA) AI models in programming through publicly available benchmarks, the research of scholars and professionals in the computer science field, as well as broader community discourse within the software engineering community to predict how AI programming models will impact the field.

Over the past decade, software engineering has quickly gone from being a niche profession to one of the most popular and high-paying career paths. In 2014, the median base salary for a SWE in the United States was over 100,000 USD. And by 2024, this figure had crossed well over 120,000 USD (Nezaj, 2024). The growth of the internet, and the migration of humanity to the digital realm, have provided an unprecedented amount of demand, and consequently pay, for individuals who can code. Today, though, the demand for software engineers has dramatically decreased (Nezaj, 2024). One primary factor for this decrease in employment is the development of AI, particularly the ability of AI to write and edit code. Evaluating this development is key to understanding the position of SWEs today, and what the future of computer science work looks like.

One of the first benchmarks evaluating publicly available LLMs' performance on coding tasks was published in the paper *Evaluating Large Language Models Trained on Code* in July of 2021 by a team at OpenAI. The objective of the paper was to fine-tune a generative pre-trained transformer (GPT)-based language model on code from GitHub (Chen et al., 2021). A GPT-based language model is an AI system that utilizes transformer blocks to process natural language inputs and generate outputs. A pre-trained model refers to a GPT model that has been trained on a sufficiently large dataset such that the model can accurately predict what word will come next in any sentence. In other words, it's a model that has a humanlike understanding of language.

A fine-tuned model is a pre-trained system that has been further trained for a specific task. For example, a GPT model trained on large chunks of text could be fine-tuned to be a conversational assistant, like ChatGPT. In *Evaluating Large Language Models Trained on Code*, the OpenAI team fine-tuned a pre-trained language model on publicly available code (Chen et al., 2021). This means that Codex, the fine-tuned model, was trained on English text and fine-tuned to understand and write code (Chen et al., 2021). The team evaluated Codex's performance on certain programming benchmarks against other SOTA models at the time, most notably GPT-3, the pre-trained version of the fine-tuned ChatGPT-3 (Chen et al., 2021). The team found that Codex performed significantly better than GPT-3 across all benchmarks tested. Most notably, the team found that, when provided a "docstring"—an empty code function and a comment on what to fill in—from their dataset, Codex was able to create a working implementation for 28.8% of their 164 hand-written test problems, while GPT-3 failed to produce a single working implementation (Chen et al., 2021). Through their experiments, the team proved that while fine-tuned GPT models had the potential to be proficient at programming, pre-trained GPT models at that time were not.

Evaluating Large Language Models Trained on Code was published in May of 2021, well over a year before the release of the first public ChatGPT model. This was a time before the world, and more specifically, software developers, understood the powerful capabilities of AI models. AI research was far less popular than it is today, and for the developers that spent their time exploring the research paper, there was no way to access Codex, let alone see the model work in action. Codex was not released as a groundbreaking AI agent or developer tool, but rather presented as a technical demonstration that may or may not come to fruition.

While Codex proved GPT-based models had potential programming capabilities back in 2021, Cognition Labs' release of their AI SWE, Devin, in March of 2024 actualized this potential. Devin used OpenAI's pre-trained GPT-4 model and fine-tuned it on code. Unlike Codex, though, Devin was fine-tuned not to respond to prompts, but to autonomously develop code given a prompt. In other words, Devin isn't a GPT chatbot that had seen a lot of coding problems, but instead is an autonomous engineer built from a SOTA foundation model. Also unlike Codex, Devin is not a research prototype, but a production-grade tool actively deployed by industry-leading companies like Origin, Hudl, and Microsoft (Devin AI, 2024).

Many people within the greater computer science/technology field have had negative reactions to the development of Devin. On an OpenAI community forum discussing the release of Devin, users fear for the future of programming jobs. User skarkonan1375 comments, "As a CS student who is graduating in a year, it terrifies me! Job market is already terrible as it is, and now I'm not sure how this is going to affect it!" Another commenter, agenttim007, questions, "After seeing something like this, I wonder if there is any point learning to code nowadays?" (OpenAI Community, 2024).

Despite the fear, many more AI coding softwares were released soon after the release of Devin in early 2024, most notably Cursor and Windsurf. These two products are both built off of Visual Studio Code, one of the most popular code editing softwares. Unlike Visual Studio Code, though, both code editors have built-in AI models that can autocomplete code. Even more notably, both softwares have terminals where the user can enter instructions for the code editor, and the software will write code based on the user's input. Unlike Devin, these tools work alongside the user, responding to user questions and suggesting code as the user works.

These tools have attracted considerable attention from software engineers globally—Cursor more so than Windsurf—with many engineers at companies like Google and OpenAI posting online about how much more efficient their coding pipeline is with these tools. Kevin Whinnery from OpenAI says, "The Cursor tab completion while coding is occasionally so magic it defies reality," while Logan Kilpatrick, senior product manager at Google DeepMind, comments, "Cursor is the best AI developer tool right now, avoid it at your own peril" (Anysphere Inc., 2023). While those who have adopted these AI tools praise their efficiency, it seems this increased efficiency presents a significant challenge to the SWE labor market. Today, any demand in the economy for software engineers can be more greatly satisfied by a single SWE with AI tools than many traditional SWEs without AI assistance. With the rollout of these AI tools, the total amount of SWEs needed for any given amount of work has decreased dramatically.

Even more threatening than today's software engineers using AI to boost productivity is the rise of proprietary AI engineers being developed by big tech companies. In an episode of *The Joe Rogan Experience*, Mark Zuckerberg explains how he sees this fear playing out at Meta, commenting:

> Probably in 2025, we, at Meta, as well as the other companies that are basically working on this, are going to have an AI that can effectively be a sort of mid-level engineer you can have at your company, that can write code. And once you have that… over time it will get to the point where a lot of the code in our apps, and including the AI we generate, is actually going to be built by AI engineers instead of people engineers. (Zuckerberg, 2024)

Zuckerberg's promise, and the general drive within the tech community to develop autonomous artificial intelligence, promises to be destructive for employment of entry-level SWEs. To understand how these technical developments will affect the job market and the lives of millions of people worldwide, I turn to Business Insider's Amanda Hoover. In her 2025 exploration *The AI Coding Apocalypse*, Hoover explains how the world of SWE is rapidly changing before our eyes. She considers the all-time poor job market for SWEs, the 50% decrease in the rate of job postings in 2023, and the all-time low postings for SWE work on Indeed today.

Hoover notes that, while the poor state of the SWE job market today can be largely attributed to an over-hiring of SWEs during the pandemic, the rise of AI coding agents is likely to exacerbate these labor market trends. She explains that experienced developers have little to worry about when it comes to AI's ability to code. AI tools are unable to complete novel, complex problems that experienced engineers are getting paid so much to solve. But, where Hoover sees AI as a helpful commodity to high-level SWEs, she views it as a substantial threat to entry-level software engineers. She reinforces Zuckerberg's sentiment on the power of mid-level AI engineers, commenting, "While more-experienced engineers are optimistic about AI, young engineers have more reason to worry" (Hoover, 2025). Hoover shares a conversation she had with Alexander Petrov, a current SWE, who explains her thinking: "[AI]…does remove the ladder upon which junior developers would try to do those things, make those mistakes, and then learn" (Hoover, 2025). Considering AI coding software can now write code as well as—if not better than—most entry-level engineers, it seems likely that tech companies will begin taking AI programmers over human ones for junior positions.

To effectively understand what the field of software engineering will look like within the coming years, I turn to the research paper *Will Artificial Intelligence Replace Software Developers?* In this paper, authors Mamdouh Alenezi and Mohammed Akour analyze how recent AI innovation will impact the software engineering field. The authors looked at how recent AI developments have been optimizing work efficiency in the software engineering field, partially by conducting their own research. They surveyed 250 participants who had more than two years of experience in the software engineering industry. The results showed that around 70% of their 250 respondents saw that using AI tools for coding was beneficial for their work efficiency (Alenezi & Akour, 2025).

The team went on to evaluate the workforce implications of the development of these models. They concluded that, given AI coding software would continue to permeate the SWE workforce, the skill gap between today's SWEs and the skills needed to effectively use AI could result in operational inefficiencies across teams. Specifically, they concluded that "many SWEs may lack the requisite AI expertise, leading to a skill gap that can impede the effective utilization of AI tools" (Alenezi & Akour, 2025). Moreover, their research anticipated resistance from professionals in adopting AI tools: "Adopting AI technologies can encounter resistance from practitioners accustomed to traditional software engineering methodologies. Concerns may include fear of job displacement, skepticism about AI effectiveness, or discomfort with altering established workflows" (Alenezi & Akour, 2025).

Alenzi and Akour concluded their paper with their vision on where they saw software engineering going, summarizing: "The convergence of AI and IoT is poised to create a new paradigm in software engineering… SWEs will need to develop systems capable of handling real-time data processing and analysis, ensuring seamless integration between AI algorithms and IoT infrastructures" (Alenezi & Akour, 2025). From this research, I conclude that the software engineering community may experience significant workforce reductions as current SWEs gain more efficiency programming with AI tools. Considering Alenezi and Akour found that a vast majority of survey respondents admitted to using AI coding tools and claimed they were more efficient because of these tools, I assume we can expect to see less work for SWEs. Because each individual AI-backed programmer can do more work in a shorter time than a traditional programmer, we can expect to see fewer programmers needed to complete a given amount of work. Within upcoming years, the software engineering industry will still have human programmers doing most of the work. There will just be a lot fewer programmers than there are now.

The computer science industry has always focused on driving technology forward. Before the industry was equipped with high-level programming languages and millions of SWEs across the world, the industry focused on pushing the capabilities of computer technology, pushing the boundaries of what we thought was possible. In the 2024 paper exploring the history and future of programming, *Beyond Automation: AI as a Catalyst for New Job Creation in Software Development,* authors Jill Willard and James Hutson take an extensive look towards the history of software development in order to predict what the future of the field entails. Willard and Hutson examine how software development has evolved from punch cards and complex electrical engineering in the 1940's and 1950's to the high-level programming languages today. In the end, Willard and Hutson concluded that, just like how high level programming languages lowered the technical bar of entry for SWEs, and promoted creativity and exploration in the field, this new wave of AI automation will once again reduce the technical depth traditionally required of SWEs, and reward developers who can make use of the AI tools at their disposal (Willard & Hutson, 2024).

Similarly to Willard and Hutson, I conclude that AI-automated code will have a major impact on decreasing the barrier of entry to develop software. Yet, I take this a step further and conclude that AI will greatly shrink the white-collar SWE workforce to a minor fraction of what it is today, if not eradicate the workforce entirely. I see the field of computer science shrinking down once again and becoming more like a group of driven scientists pushing the limits of what we think computer technology can do. Much of the work software engineers perform today—developing websites, applications, and games—represents an intermediate step in the development of human technology. Humans writing code is not, and was never meant to be, a permanent method of creating what we want in computer systems, but rather an intermediate step in our journey towards seamless interaction with computers.

Just as computer scientists in the 1990s worked on creating programming languages so we could more easily write software, the computer scientists of today are writing AI coding agents so we can more easily write software using natural language prompts. In this sense, this intermediate "wave" of time—where we had millions of software engineers across the world writing code—is going to crash. And, as we progress through time, our field will continue to ride these waves between major advancements in computer technology, crashing as we perpetually simplify the interactions between humans and machines.

References

Alenezi, M., & Akour, M. (2025). *Will artificial intelligence replace software developers? Applied Sciences, 15*(3), 1344. https://www.mdpi.com/2076-3417/15/3/1344

Anysphere Inc. (2023). *Cursor: The AI code editor*. https://www.cursor.com/

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code (arXiv:2107.03374). arXiv. https://arxiv.org/pdf/2107.03374

Cognition. (2024, March 12). *Introducing Devin, the first AI software engineer* [Video]. YouTube. https://www.youtube.com/watch?v=fjHtjT7GO1c

Devin AI. (2024). *Devin*. https://devin.ai/

Hoover, A. (2025, February 25). *The AI coding apocalypse*. *Business Insider*. https://www.businessinsider.com/career-ladder-software-engineers-collapsing-ai-google-meta-coding-2025-2

Nezaj, J. (2024, June 17). *The rise and fall of the software developer*. ADP Research Institute. https://www.adpresearch.com/the-rise-and-fall-of-the-software-developer

OpenAI Community. (2024, March 22). *Fully autonomous AI software engineer – Devin* [Online forum post]. OpenAI Community Forum. https://community.openai.com/t/fully-autonomous-ai-software-engineer-devin/679804/25?page=2

Willard, J., & Hutson, J. (2024, August). *Artificial intelligence and the software developer labor market* [Faculty research paper]. Lindenwood University Digital Commons. https://digitalcommons.lindenwood.edu/cgi/viewcontent.cgi?article=1675&context=faculty-research-papers

Zuckerberg, M. (Guest). (2024, January 15). *Joe Rogan Experience #2255 – Mark Zuckerberg* [Audio podcast episode]. In J. Rogan (Host), *The Joe Rogan Experience*. YouTube. https://www.youtube.com/watch?v=7k1ehaE0bdU