

How Artificial Intelligence Is Reshaping the Work and Career Paths of Junior Software Developers in China and the United States

Yuxiang Zou

*Villa Madonna Academy, Villa Hills, USA
samzou16@outlook.com*

Abstract. The widespread adoption of generative AI in software development can impact junior developers as AI tools can help with coding, testing, debugging, and documentation tasks. This paper will explore how AI influences the job, the necessary skills, and the career paths of junior software engineers, and why this can vary across different countries, such as China and the USA. Experiments, job-vacancy studies, surveys, government statistics, and industry evidence are the bases for the analysis. While results have been inconsistent for different tasks and contexts, research indicates that AI can boost developer productivity, with one field experiment finding it has a greater impact on less-experienced developers. There is more code being written, more code to be reviewed, and more code to be communicated and understood because of AI's increasing automation of repetitive tasks and coding. While software developers should still be in demand, competition for jobs in this field might increase. The direct evidence of changes in junior hiring is more pronounced in the United States, while China has a massive developer community and a growing domestic AI industry. In conclusion, the evidence points to the evolution of the junior developer position, not its demise, due to the influence of AI.

Keywords: Artificial Intelligence, Junior Software Developers, Skill Requirements, Career Development, Labor Market

1. Introduction

Generative AI has rapidly become integrated into software development. According to the Stack Overflow 2025 Developer Survey, 84 percent of respondents were using or intending to use AI tools, and 51 percent of professional developers said they use them on a daily basis [1]. Meanwhile, 46 percent did not believe AI-generated content was accurate, while 33 percent did [1].

Junior software developers may be especially affected. The term "junior" is used as a general classification of early-career, and no single definition of the term was used across the studies. The labor-market analysis is based on the junior and senior vacancy categories as defined by Westby et al. [2].

It is aimed at standard, less critical tasks like smaller features, testing, debugging, maintenance, and documentation. These are also tasks that new developers get experience in. In a survey of 481

programmers, Sergeyuk et al. investigated the use of AI for feature implementation, writing tests, triaging bugs, refactoring code, and documenting it in natural language [3]. A developer receiving a first draft from AI needs a somewhat different mix of skills from someone writing code manually. Moreover, as AI takes over more mundane tasks, it could impact the way junior developers gain experience.

This paper addresses the following two questions. First, how is AI changing the daily work and skill requirements of junior software developers? Second, why could these changes take different forms in China and the United States? It also takes into account whether AI alters the trajectory for junior developers to become more senior.

Much of the research that has been carried out has concentrated on productivity, skills or employment. There are few studies that relate task changes to the requirements for skills, entry-level hiring, and career development. Another reason for the difficulty in making the China-U.S. comparison is that the two countries do not publish fully comparable statistics for junior software developers. Practically, there are still some questions yet to be answered: What skills will new developers still need, and how will they continue to build experience in the age of AI?

The analysis of this relationship between tasks, skills, and career development is followed to answer these questions. AI first alters the way that development tasks are done. Those changes impact the skills that employers are looking for and can impact entry-level hiring and the development of junior developers. The impacts occur within various labor markets and technology ecosystems in China and the United States.

2. Literature review

2.1. AI and developer productivity

While recent research has frequently shown that AI coding assistants boost productivity, the magnitude and direction of this impact vary depending on the work being assessed. Clearly defined programming tasks often yield substantial gains, whereas studies of mature codebases, collaborative work, or later-stage production report smaller gains or additional costs.

Peng et al. tested an HTTP server in the JavaScript language in a controlled experiment [4]. Participants using GitHub Copilot completed the task 55.8 percent faster than the control group. The review, integration, maintenance and release were not measured, so it is not a measure of end-to-end development, but is useful in a narrow coding task.

Evidence is provided by Cui et al. in actual workplaces [5]. Their randomized field experiments at Microsoft, Accenture and an anonymous Fortune 100 company featured 4,867 developers. Overall, the number of completed tasks rose by 26.08 percent with the use of an AI coding assistant, and less-experienced developers indicated higher use and higher productivity gains. AI could be particularly helpful to junior developers, especially for information retrieval, syntax, and typical implementation steps.

When the developers are familiar with a codebase, another conclusion comes about. Becker et al. examined 16 professional open-source developers working on 246 tasks in their respective repositories, with an average of 5 years of experience working on them [6]. The developers spent 19 percent longer with the AI tools in early 2025. The differences in developers, tasks, and environments prevent the direct comparisons of the results; prompting and verifying the AI suggestions may have generated additional work rather than saving it.

In the working paper that was subsequently updated, Song et al. found that GitHub Copilot increased project-level productivity by 6.5 percent, individual productivity by 5.5 percent, and

participation by 5.4 percent in open-source projects [7]. Integration time, however, went up by 41.6 percent. Even though more code was generated, integration was even longer.

In 2025, DORA's research also reveals a similar organizational trend [8]. Drawing on the insights of nearly 5,000 tech professionals and over 100 hours of qualitative research, it defines AI as an amplifier – good development processes can help teams leverage AI, and poor processes can make them more apparent as the rate and volume of software changes accelerates [8]. Given that DORA is primarily based on survey and qualitative evidence, these relationships should be interpreted as associations rather than direct causal relationships [8].

In all of these research studies, AI seems to be most helpful with information retrieval or well-defined tasks. When it relies on project knowledge, coordination, review and integration, its benefit is less certain. This is important because early-career work may contain more 'routine' work, and senior work will have more responsibility for the broader system.

2.2. AI and skill changes

Analysis of developer skills reveals a similar trend: AI can simplify coding, but it also makes it crucial to evaluate whether coding results are accurate, valuable, and safe.

In 2025, Sergeyuk et al. conducted a survey of 481 programmers to find out about their experiences with AI support for implementing features, writing tests, debugging, refactoring, and documentation in natural language [3]. Use was constrained by limited trust, restrictive company policies, and insufficient project context, although developers commonly delegated repetitive tasks to the tools. The key question was whether such output could be relied upon in a real project.

On a broader scale, the 2025 Stack Overflow Developer Survey reveals the same challenge [1]. The majority of respondents who answered the question (66%) cited AI solutions that were "almost right" as a major annoyance, and 45% said that debugging AI-generated code can be more time-consuming [1]. A junior developer can be handed believable code in no time at all and then lack the technical knowledge to determine if it's functional.

To explore the gap between output and comprehension, Martin-Lopez et al. examined the difference between output and understanding in a programming experiment involving 69 participants [9]. While those who used AI were able to complete over twice the median level of tasks, they were 12.5 percent less successful in answering technical questions on the code. The experiment does not establish long-term skill loss, but it does demonstrate that there are different outcomes for the production of more complete code versus understanding of the code.

Weisz et al. analyzed 669 users of an internal IBM AI code assistant and saw inconsistent productivity gains, and questioned its ownership and responsibility for code [10]. The responsibility for the performance, security, and project requirements of the AI-assisted software still falls on the shoulders of the developer and/or organization.

According to the Future of Jobs Report 2025 by the World Economic Forum, the World Economic Forum's Future of Jobs Report 2025 reports that employers expect 39 percent of workers' core skills to change by 2030 [11]. Analytical and creative thinking skills are important, as are the fast-growing technical skills, such as AI and big data [11]. Technical knowledge is still required, but review, problem-solving, communication and AI evaluation are increasing in importance.

2.3. AI and entry-level employment

Changes in employment are more difficult to measure than changes in productivity. Junior technology job postings did not necessarily decline as a result of AI, as this coincided with a general

slowdown in the technology sector in 2022.

The most direct recent evidence is provided by Westby et al. [2]. They estimate a relative drop of 14 to 15 percent in the number of junior compared to senior software-developer vacancies in the United States in the wake of ChatGPT's public release using near-universe U.S. online vacancy data from Lightcast. They also discover that increased experience levels were largely a result of employers demanding increased experience in the same job title as opposed to changing the job title from a lower to a higher level.

The other junior positions had a higher weight on problem solving, communication and attention to detail than on AI. Employers seem to be looking for more than just the use of AI tools from entry-level developers.

The study has not yet been published in a peer-reviewed journal, but is an IZA Discussion Paper. Event-study and differences-in-differences methods are employed, and software development is compared to other professions, adding to the analysis. However, this 14-15 percent figure should be interpreted more as a relative movement than an absolute number of jobs lost solely to AI.

Humlum and Vestergaard offer more general labor market findings [12]. Using AI-adoption surveys and Danish administrative data, they find widespread adoption and task changes but little evidence of large average effects on earnings or working hours during the first two years.

According to the 2026 report by the World Economic Forum (WEF), more than one-third of the world's young workers are employed in jobs that have medium to high exposure to the task change driven by AI, with a particular emphasis on job access, job design, education-system alignment, and talent pipelines [13]. The number of positions available is not the only question for junior software developers. Firms should also look at how new developers will acquire the knowledge and judgment that have historically come from doing the repetitive, less-risky work if AI takes it over.

2.4. China, the United States, and the research gap

There is an asymmetry in the evidence available to compare China and the U.S. The most robust evidence on hiring juniors in software is drawn from the U.S., and Chinese evidence is more likely to be regarding software in general, the demand for AI labor, the software developer community, and adoption of AI tools. To make a direct comparison between the two evidence bases would overestimate the conclusions possible from the data.

According to the Bureau of Labor Statistics, in 2025 there were approximately 1.718 million software developers in the United States, and the median annual salary for software developers was \$135,980 [14]. The U.S. Bureau of Labor Statistics estimates that employment will increase by approximately 10 percent in the 10-year period ending in 2035 [14]. These numbers refer to the occupation overall, and so overall growth is possible despite a more "selective" entry-level market.

China has larger units of measurement. According to a report by the National Data Administration, in 2024, there were more than 9.4 million software developers, and China was the second-largest community of open-source participants in the world [15]. Thus, due to this difference between the occupation count in this measure and that used by the U.S. BLS, a simple ratio of the two should not be used.

The software industry in China also continued to grow. In 2025, the revenue of the software and information-technology service industries reached 15.4831 trillion yuan, representing a 13.2 percent increase, and the revenue of information-technology services reached 10.6366 trillion yuan, an increase of 14.7 percent [16]. These are industry growth data, not data on junior hiring.

There is recent academic evidence from China provided by Fang et al. [17]. By combining information from 1.6 million online job postings from 2016 to 2024 and household data, they find

that stronger local demand for AI-related labor is associated with positive but uneven wage effects across regions and groups. The study is not specific to junior software developers, but demonstrates that AI employment impacts depend on local factors and supporting infrastructure.

Kulikova and Tai report a different pattern of AI-tool adoption among Chinese developers: GitHub Copilot is less common than in the global sample, while other tools, including Chinese alternatives, are more prominent [18]. The survey was limited in the range of tools available, and so should not be viewed as a comprehensive survey of the Chinese market.

There are also significant disparities in AI investment. According to the 2026 AI Index report by Stanford, according to Stanford's 2026 AI Index, measured private AI investment reached \$285.9 billion in the United States in 2025, compared with \$12.4 billion in China [19]. This might be an underestimate of China's overall AI spending, as government guidance funds are not included in the same manner [19]. The figures have significantly more private investment measured in the United States and a different ratio of total AI investment in the United States.

The primary disconnect is between productivity and task shifts and early-career development in labor markets. Direct evidence on junior hiring is substantially stronger for the United States than for China. In this paper, therefore, it is not assumed that one has a set of symmetrical data.

3. Research method

A comparative secondary-data research approach is used, using experiments, job-vacancy analysis, developer surveys, government statistics, and select industry sources. Peer-reviewed studies and government statistics are given first preference; working papers, surveys and company reports are given less credence. Since China and the United States have different definitions of statistics, the numerical statistics are not considered to be directly comparable.

4. Theoretical foundation

4.1. Task-based theory

Task-based theory can be used to explain how AI can transform a job without eliminating the occupation itself. Acemoglu and Restrepo characterize production as a collection of activities that can be carried out by either labor or technology [20]. Automation creates a displacement effect when technology performs tasks previously completed by workers; technological change can create new jobs or make activities more important in which people still have an advantage.

This is a good fit for software development. A developer does not do one thing – coding. The role also involves comprehending requirements, testing, debugging, reviewing changes, communicating with colleagues, merging parts, and owning the product put out. Some of these tasks are easier for AI to help with than others.

Acemoglu takes a similar task-based perspective on modern AI [21]. He views the impact of AI overall as a combination of the percentage of tasks impacted and the magnitude of the productivity gain on those tasks. Good performance on easy skills does not necessarily mean good performance on more contextual and judgmental skills.

Task-based theory suggests that in the case of junior developers, the amount of work done will shift but not be replaced. Requirements analysis, integration, review, and final responsibility are more of a responsibility for humans, but first-draft coding, test generation, and documentation are more open to assistance from AI.

4.2. Human capital theory

Task changes are important for career development and are explained by human capital theory. Becker views education and training as investments that help to develop productive knowledge and skills [22]. This involves work experience as workers continue to learn after formal education.

This type of learning has always been a characteristic of junior software jobs. For the new developer, it could start with bug fixes, testing, maintenance or just features. Successful repetitions using real systems make developers familiar with codebases, failures and team decisions.

AI can impact this learning process in opposing manners. It can facilitate learning and explain new code, examples, or even assist a junior developer in getting started on a new framework. This possibility is supported by the finding from Cui et al. that less-experienced developers had a greater productivity boost [5]. If developers repeatedly accept AI-generated code without understanding it, AI may instead reduce opportunities for learning. This danger is seen in lower technical understanding, as reported by Martin-Lopez et al. [9].

These theories entail two expectations. First, AI should primarily shift the balance of work within the junior software engineer's position, not move all aspects of the role into the AI domain. Secondly, career development should be impacted only if the work with AI does not provide chances for junior developers to acquire knowledge and judgment. The learning path for juniors to develop expertise can become weaker if AI takes away the learning opportunities in routine work, but does not replace them. If AI takes them out of the "understand" part and puts them in the "hard" part, it could speed up human-capital development.

5. Data and mechanism analysis

5.1. Changes in daily work

The first mechanism is a change in workflow. In a traditional way, an application is developed by a junior developer, who reads a requirement, writes code, executes tests, discovers errors, makes changes to the code, and documents it. At present, AI can be deployed in various phases of this process. A developer can ask for a first version, develop test cases, inquire about an explanation of an error or develop a first version of documentation. As demonstrated by Sergeyuk et al., AI assistance is already being implemented in many of these tasks by developers [3].

Table 1. How AI changes common junior software development tasks

Task	Traditional workflow	AI-assisted workflow	Human responsibility that remains
Coding	Developer writes most code manually	AI generates or completes sections	Decide whether the solution fits the system
Testing	Developer designs and writes tests	AI drafts test cases	Judge coverage and interpret failures
Debugging	Developer searches and traces errors	AI suggests explanations and fixes	Verify the cause and test the correction
Documentation	Developer writes summaries and comments	AI drafts documentation	Check accuracy and project context

Note. Author's synthesis based on Sergeyuk et al., Stack Overflow, and DeBellis et al. [1, 3, 8]. The first step is a change from the creation of a first answer to the checking, correcting and integrating of a first answer. Stack Overflow's findings on nearly correct AI outputs underscore the continuing need for verification and correction [1].

Demirer et al. analyze more than 500,000 GitHub developers [23]. In the September 2026 update, they estimate that autocomplete, interactive coding agents, and autonomous coding agents boost commit activity by 30, 180 and 240 percent, respectively. But for autonomous agents, the 240 percent rise in commits drops to 80 percent for projects and 30 percent for releases.

The difference between the commits and releases illustrates that better code generation does not necessarily mean more product software. There is still review work, testing, integration, and other downstream work to be done. Demirer et al. employ a matched event-study design, rather than random assignment; their estimates should thus be treated with some caution [23].

A similar trend is observed by Song et al.: a project-level productivity boost of 6.5 percent and an integration time boost of 41.6 percent [7]. This transition might accelerate review and integration into the job for junior developers. A junior developer might get a first draft sooner rather than spending most of a task manually writing code, and then more time determining if it works and is compatible with the rest of the system.

5.2. Changes in skill requirements

The second mechanism concerns change in skill requirements. If AI can generate a rough draft in a shorter amount of time, developing the skill of coding routine content faster might not be the most unique, but analyzing the code might be more important. There is still a need for coding skills, as one must have the technical expertise to review, debug and integrate the output and determine if it should be used or not.

This shift is evidenced in the labor market by Westby et al. [2]. The junior vacancies that remained after ChatGPT's release placed greater relative emphasis on problem-solving, communication, and attention to detail rather than AI-specific skills. Employers seem to be looking for abilities that help them grasp issues and make judgments of outcomes rather than just being able to use an AI tool.

Problem-solving is important because a model may generate technically feasible programs which do not meet the requirements. Requirements are the source of communication; they come from users, managers, designers and other developers. Speeding up code generation is of little use if it is not ensured that the code is generated for the correct goal.

For a similar reason, code review becomes important. Martin-Lopez et al. demonstrate that when completing a programming task, participants with AI assistance can execute more, but don't necessarily exhibit the same understanding [9]. The principles of programming are still required. Understanding algorithms, data structures, debugging, architecture and security provides developers with a foundation to evaluate the output from AI. The new set of skills is not less technical.

5.3. Entry-level hiring and career paths

The third mechanism relates to the recruitment aspect. Increased productivity per developer can lead to a decrease in the need for roles focused primarily on repetitive tasks, as well as increased productivity of junior hires. The net impact on entry-level jobs is a consequence of the substitution and augmentation.

This is seen in the case of the United States. The Bureau of Labor Statistics predicts that the software-developer job market will experience 10 percent growth from 2025 to 2035, and Westby et al. estimate a 14 to 15 percent relative decline in junior versus senior software-developer vacancies after ChatGPT's public release [2, 14]. These patterns can coexist: the occupation may expand overall even as employers become more selective when hiring entry-level workers.

However, causal interpretation requires caution because the adoption of AI coincided with a broader slowdown in technology hiring. Comparing the numbers of people hired in different occupations, while using econometric controls, as Westby et al. do, can still only be evidence of a relative shift in junior hiring, not a direct count of how many jobs were lost due to AI [2].

Two effects may take place concurrently. Substitution can lead to a decrease in the demand for employment focused on routine tasks, and augmentation can increase the productivity of hired juniors. Cui et al., for instance, report that the productivity benefits are more pronounced for less experienced developers [5]. There is currently no evidence of a predominant effect across firms or projects.

The chronic issue is learning, as companies need experienced developers. Talent pipelines are one of the key problems in AI-exposed entry-level work highlighted in the 2026 report from the World Economic Forum [13]. If routine tasks are reduced, companies may need more intentional mentorship, project exposure, code review, and production responsibility to provide junior developers the experience they need to develop the judgment required for senior roles.

The career trajectory can thus alter. Junior developers can spend less time writing repetitive boilerplate code and more time learning to review, test, explain, and integrate AI-generated code. Internships, project work, mentoring, and production experience may gain increased significance, as AI cannot automatically provide these forms of situated learning to new developers.

5.4. Why china and the United States differ

Although the technology is similar between the two countries, the market conditions are not.

Software development is a big and well-paying job in the U.S. According to the Bureau of Labor Statistics, there are approximately 1.718 million software developers in 2025, and the median software developer annual salary is \$135,980 [14].

Because developer compensation is relatively high, even a small productivity increase can be of economic value to an employer.

Another aspect of this context is U.S. private investment in AI. According to the 2026 AI Index report by Stanford, the total estimated private AI investment in the U.S. for 2025 is \$285.9 billion, while the investment in China stands at \$12.4 billion [19]. This comparison might underestimate China's overall investment in AI since certain government guidance funds are not reported in the same manner [19]. The data do not provide a precise ratio for total AI spending, but rather a difference in private investment that is measured.

China's software market has a different scale and structure. In 2024, the National Data Administration estimated there were over 9.4 million software developers, but this is not directly comparable with the U.S. BLS count [15]. In 2025, the Ministry of Industry and Information Technology reported that software-business revenue grew 13.2 percent to 15.4831 trillion yuan and information-technology service revenue rose 14.7 percent to 10.6366 trillion yuan [16]. These figures describe the scale and growth of the broader software industry rather than junior hiring.

Fang et al. provide complementary evidence on the broader labor market [17]. Drawing on 1.6 million online job postings between 2016 and 2024 and household data, they discover that higher local demand for AI jobs is linked to positive but regionally and group-wise variable wage effects. Though the study is not specific to the labor market outcomes of junior developers, it demonstrates that local conditions and complementary infrastructure are linked to such outcomes.

Developer-tool use is also an indication of these conditions. Kulikova and Tai note that there was a different trend in using AI tools compared with the globally surveyed population of developers and that the list of tools was restricted [18]. This is another example of Alibaba Cloud's Qoder CN [24].

Formerly known as TONGYI Lingma, Qoder CN emphasizes Chinese models, domestic deployment, and data-security and compliance requirements [24]. These features are examples of variations that local context can cause in the products of AI coding.

Table 2. Evidence relevant to the China-U.S. comparison

Dimension	United States	China
Evidence on junior hiring	Direct software-vacancy evidence is available	Comparable junior-level vacancy evidence remains limited
Broader developer market	Large, high-wage occupation with projected long-term growth	Very large reported developer ecosystem and expanding software industry
AI investment and ecosystem	Much higher measured private AI investment	Large domestic AI ecosystem, with private-investment data capturing only part of total spending
Local AI-development context	Strong commercial market for AI-assisted development	Domestic models, local deployment, and compliance requirements play a visible role
Main evidence limitation	Junior vacancy changes cannot be fully separated from the wider technology slowdown	Industry and AI-market evidence is stronger than direct junior career-path evidence

Note. Author's synthesis based on Alibaba Cloud, Fang et al., Ministry of Industry and Information Technology of China, National Data Administration of China, Stanford Institute for Human-Centered Artificial Intelligence, U.S. Bureau of Labor Statistics, and Westby et al. [2, 14-17, 19, 24].

Given the wider evidence in China, the comparison is stronger in terms of mechanisms than in directly comparable employment estimates. The technology change is working in two different markets. U.S. evidence reflects significant investment in and productivity returns from AI as well as evidence of a more targeted hiring process, while Chinese evidence reflects extensive software activity and a burgeoning domestic AI ecosystem, but less clarity regarding junior hiring.

6. Conclusions

Current research does not support a simple replacement story for junior software developers. AI is altering the landscape of work that needs to be done by humans. Coding assistants can save time in writing initial draft code, testing, documentation, and some debugging. However, code production alone does not necessarily translate to an equal increase in software product, with review, integration, verification and responsibility all still requiring human effort.

The skills that are required will shift as the work evolves. Programming fundamentals remain necessary for junior developers and are increasingly applied to assessing and correcting AI-generated work. When a developer is required to make decisions about whether or not an AI-generated solution aligns with a project, problem-solving, communication, attention to detail, and code review become crucial.

The impact on career development is less clear. Despite the positive official forecast for software-development jobs through the long term, U.S. vacancy research reveals that entry-level software hiring is getting more competitive. AI can also significantly boost productivity for less-experienced developers. Thus, AI may reduce some routine entry-level work while helping the junior developers who are hired become productive more quickly.

AI features like this are moving into other labor markets. In the United States, the direct evidence for vacancies at the junior level is better, and the amount of private investment in AI is higher and measured differently. There is a very large developer ecosystem in China, but less direct evidence of junior hiring and promotion in the software industry.

The main limitation is the lack of long-term and directly comparable junior-level data, especially for China. A number of findings are also derived from working papers, surveys or short experiments. Future research needs to track new developers over a number of years and to look at the different aspects of hiring, technical learning, project responsibility, mentoring and promotion. This study might indicate whether AI can assist junior developers in advancing to higher levels of development in a shorter period or if it eliminates some of the more traditional learning activities.

References

- [1] Stack Overflow. (2025). 2025 developer survey. *Stack Overflow*. <https://survey.stackoverflow.co/2025/>
- [2] Westby, S., Modestino, A. S., & Cheng, P. (2026). Generative AI and the redefinition of entry-level software work (IZA Discussion Paper No. 18723). IZA Institute of Labor Economics. <https://www.iza.org/publications/dp/18723/generative-ai-and-the-redefinition-of-entry-level-software-work>
- [3] Sergeyuk, A., Golubev, Y., Bryksin, T., & Ahmed, I. (2025). Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology*, 178, 107610. <https://doi.org/10.1016/j.infsof.2024.107610>
- [4] Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2302.06590>
- [5] Cui, K. Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2026). The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers. *Management Science*. Advance online publication. <https://doi.org/10.1287/mnsc.2025.00535>
- [6] Becker, J., Rush, N., Barnes, E., & Rein, D. (2025). Measuring the impact of early-2025 AI on experienced open-source developer productivity [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2507.09089>
- [7] Song, F., Agarwal, A., & Wen, W. (2024). The impact of generative AI on collaborative open-source software development: Evidence from GitHub Copilot [Preprint, version 1]. arXiv. <https://arxiv.org/abs/2410.02091v1>
- [8] DeBellis, D., Storer, K., Harvey, N., Beane, M., Edwards, R., Fraser, E., Good, B., Kalliamvakou, E., Kim, G., Maxwell, E., D'Angelo, S., Inman, S., Murillo, A., & Villalba, D. (2025). DORA 2025 state of AI-assisted software development report. Google. <https://research.google/pubs/dora-2025-state-of-ai-assisted-software-development-report/>
- [9] Martin-Lopez, A., Tufano, R., Guglielmi, E., Sánchez, A. B., Sanz, A. L., Scalabrino, S., Oliveto, R., Segura, S., & Bavota, G. (2026). More code, less understanding? On the impact of AI assistants on developers' productivity and code ownership. *IEEE Transactions on Software Engineering*, 52(7), 2173–2187. <https://doi.org/10.1109/TSE.2026.3679627>
- [10] Weisz, J. D., Kumar, S. V., Muller, M., Browne, K.-E., Goldberg, A., Heintze, K. E., & Bajpai, S. (2025). Examining the use and impact of an AI code assistant on developer productivity and experience in the enterprise. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Article 673, pp. 1–13). Association for Computing Machinery. <https://doi.org/10.1145/3706599.3706670>
- [11] World Economic Forum. (2025). The future of jobs report 2025. World Economic Forum. <https://www.weforum.org/publications/the-future-of-jobs-report-2025/>
- [12] Humlum, A., & Vestergaard, E. (2025). Still waters, rapid currents: Early labor market transformation under generative AI (NBER Working Paper No. 33777). National Bureau of Economic Research. <https://doi.org/10.3386/w33777>
- [13] World Economic Forum. (2026, June 22). Artificial intelligence and the future of entry-level work: A framework for safeguarding and reinventing early career pathways. *World Economic Forum*. <https://www.weforum.org/publications/artificial-intelligence-and-the-future-of-entry-level-work-a-framework-for-safeguarding-and-reinventing-early-career-pathways/>
- [14] U.S. Bureau of Labor Statistics. (2026). Software developers, quality assurance analysts, and testers. *Occupational Outlook Handbook*. <https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm>

- [15] National Data Administration of China. (2025). *Digital China development report 2024*. National Data Administration of China. <https://www.nda.gov.cn/sjj/swdt/xwfb/0530/ff808081-96b466bd-0197-2084f21d-0565.pdf>
- [16] Ministry of Industry and Information Technology of China. (2026, January 30). *Operation of China's software industry in 2025*. Ministry of Industry and Information Technology of China. https://www.miit.gov.cn/gxsj/tjfx/rjy/art/2026/art_65a12a560865432bb1548fdddc74f19c.html
- [17] Fang, T., Lin, C., & Liu, Q. (2026). AI and labor market outcomes: Evidence from China. *Journal of Productivity Analysis*, 65, Article 28. <https://doi.org/10.1007/s11123-026-00817-2>
- [18] Kulikova, S., & Tai, C.-F. (2026). Insights into China's developer landscape: Key trends from the JetBrains Developer Ecosystem Survey 2025. *JetBrains Research*. <https://blog.jetbrains.com/research/2026/01/insights-into-china-s-developer-landscape-key-trends/>
- [19] Stanford Institute for Human-Centered Artificial Intelligence. (2026). The 2026 AI index report. *Stanford University*. <https://hai.stanford.edu/ai-index/2026-ai-index-report>
- [20] Acemoglu, D., & Restrepo, P. (2019). Automation and new tasks: How technology displaces and reinstates labor. *Journal of Economic Perspectives*, 33(2), 3–30. <https://doi.org/10.1257/jep.33.2.3>
- [21] Acemoglu, D. (2025). The simple macroeconomics of AI. *Economic Policy*, 40(121), 13–58. <https://doi.org/10.1093/epolic/eiae042>
- [22] Becker, G. S. (1994). *Human capital: A theoretical and empirical analysis, with special reference to education* (3rd ed.). University of Chicago Press.
- [23] Demirer, M., Musolff, L., & Yang, L. (2026). Writing code vs. shipping code: Productivity effects across generations of AI coding tools (NBER Working Paper No. 35275, revised September 2026). National Bureau of Economic Research. <https://doi.org/10.3386/w35275>
- [24] Alibaba Cloud. (2026). Qoder CN series: Product introduction. *Alibaba Cloud*. <https://docs.qoder.cn/en/product-overview/introduction-of-qodercn>