karat^

AI and the future of interviewing

April 29, 2025

Measurement is an inherently scientific domain. Measuring talent requires a deliberate approach and methodology that produces an actionable hiring signal. This signal must answer a seemingly simple question: does the candidate have the skills needed to do the job? Today, the job of software engineers is changing thanks to Artificial Intelligence. As a result, talent measurement must evolve to control for candidates using AI in interviews.

The following paper examines how the rapid proliferation of AI is changing software development and how that impacts the scientific measurement of tech talent.

Readers will walk away with a better understanding of how enterprise software leaders are integrating AI into their workflows, the evolving skills and responsibilities of software engineers, and how to update hiring processes to not just detect and prevent the use of AI, but to assess how candidates will perform in an AI-augmented world.

How AI is changing software development

When it comes to real-world AI implementations, employees are moving faster than enterprises.

At the enterprise level, agentic-AI initiatives are rapidly accelerating from experimentation to piloting, but few deployments have yet to take hold at scale. According to KPMG, the biggest obstacles to more widespread agentic deployment is risk management, data quality, and trust in the technology.

Those roadblocks are less concerning to employees, however. An overwhelming majority of software developers self-reported using AI on the job in the last year. Productivity tool usage on a daily basis is up to 58% from 22%. Knowledge assistant usage on a weekly basis is up to 61% from 48%, as is GenAI usage embedded into existing workflows, jumping from 24% to 35% (per KPMG).

For software teams, this is manifesting in streamlined documentation, improved speed, and quality. The most common AI augmentations in software development include quickly producing first-draft code generation for proof of concept projects and incubators. QC is also experiencing a significant boost from AI, where lower-level QC engineers can leverage the technology to dramatically improve the speed and quality of their work.

And while software hiring remains down from its 2021 peak, the demand for AI skills is growing. More software leaders expect to see their headcounts rise than fall over the next three years as a result of AI.

Top AI talent is moving faster than ever, often receiving multiple competing offers. The average time from invitation to interview in the U.S. declined from 12 to 10 days last year (for non-Karat-clients). Top-performing organizations completed the process in an average of 6 days.

The net result is a still-competitive market for hiring top software talent. Speed remains a competitive differentiator, but complexity is growing. The twin objectives of measuring new AI-related skills and maintaining interview consistency are forcing organizations to re-examine their hiring practices.

The impact of AI and the future of talent measurement

Amazon's recent crackdown on candidates using AI and the rise of interview coding tools have brought the topic of interview integrity to the forefront.

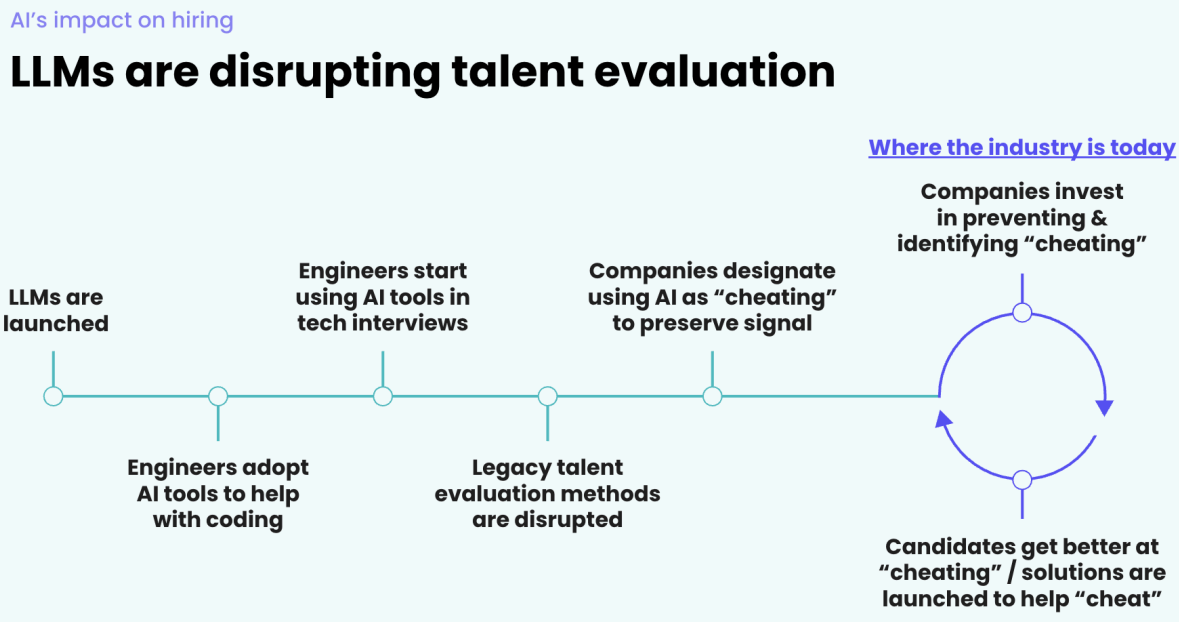
Today, most Fortune 500 companies share Amazon's point of view. However, there's a growing number of companies that are embracing the use of AI in interviews. They want to hire engineers who can leverage new tools to enhance their skills.

On the job, Goldman Sachs has given many of its engineers access to AI coding assistants like GitHub Copilot and Gemini. The company has even held competitions to foster creative AI use among developers. But that acceptance has not, yet, translated to the interview.

Whether you decide to allow candidates to use AI or not, it's essential to enforce the existing rules. The most important factor in talent measurement is consistency, which requires controlling for as many variables as possible. AI usage is one such variable. If the goal is to obtain an AI-free signal of a candidate's abilities, LLM usage must be prohibited for all candidates to ensure fairness. Companies taking this approach should explicitly state that AI use is prohibited, and have a methodology for preserving the integrity of their hiring signal.

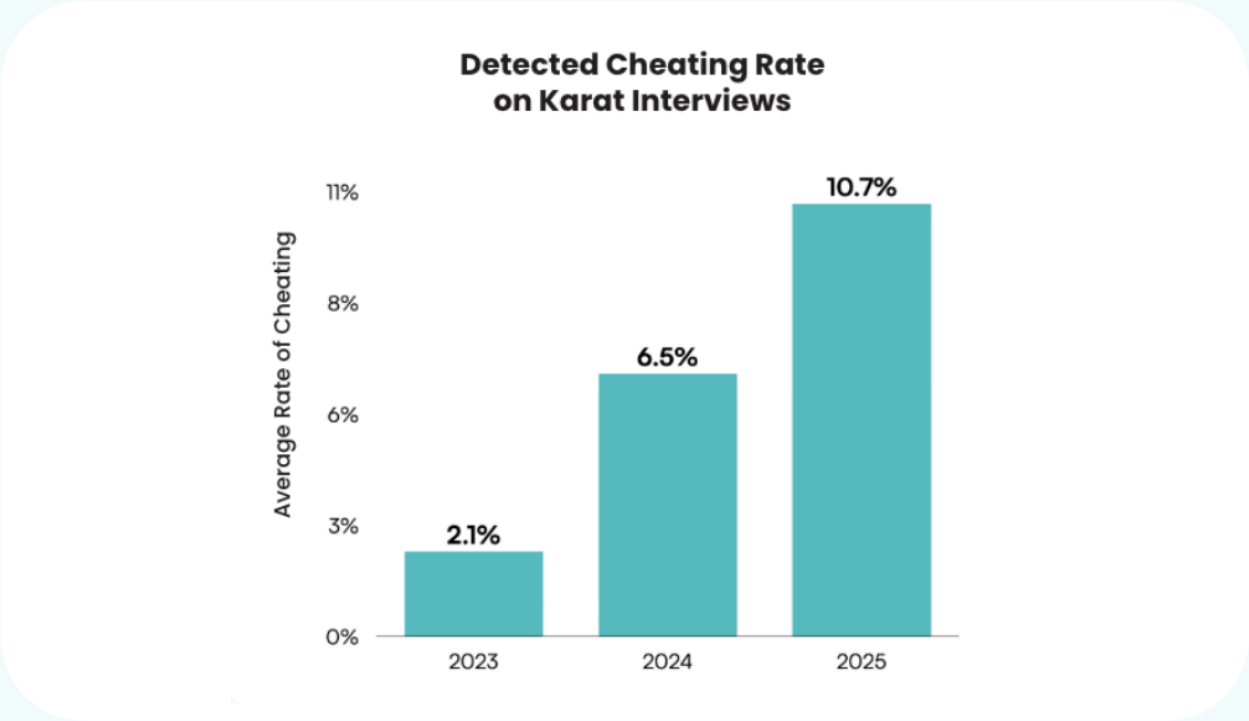
Ensuring interview integrity

Any take-home assessment without a live component is vulnerable in today's world. Coding tasks are particularly easy for LLMs to complete. One tech leader we spoke to recently shared that they suspect more than 80% of their India-based candidates were using LLMs on top-of-funnel code tests—despite being explicitly told not to. Other recent experiments conducted by 3rd parties have put that number closer to 100%.



This is why we recommend leaning into live interviews. If a candidate attempts to use LLMs despite clear restrictions, human interviewers can serve as an effective safeguard. Karat's interview engineers are taught to engage candidates in discussion, probe their understanding, and detect unauthorized AI usage.

But even with the more sophisticated detection methods of live interviewing, we're seeing candidate behaviors shift as AI becomes a more integral part of the developer ecosystem. The result is a five-fold increase in cheating detection rates over the past two years.



Types of cheating and how to mitigate them

The most common attempts at cheating in interviews fall into one of two categories:

1. Impersonation

Candidates hiring 3rd parties to take their technical interviews is nothing new, but the rise of remote work has removed the simplest safeguard: in-person interviews.

This has been a consistent problem for offline coding tests and take-home projects because ensuring a candidate is producing their own work is nearly impossible without a time-consuming, multi-step workflow that constantly re-examines a candidate's prior submissions.

Recording video interviews mitigates this risk without the extra steps by making it easier to verify a candidate's identity. While the growing prominence of AI-generated

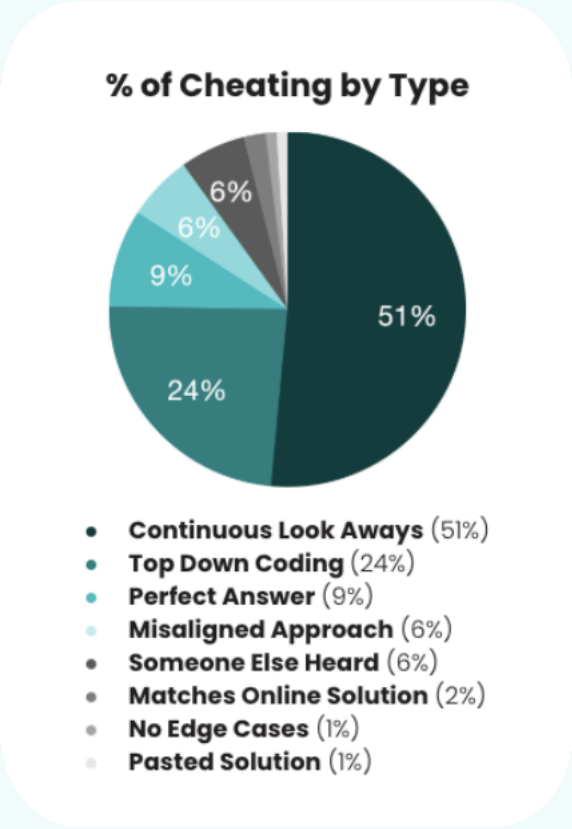
fake candidates has raised fresh concerns, the current state of deepfake technology is not advanced enough to overcome some simple verification techniques. These include asking candidates to turn off background filters, tilt their heads sideways, or raise a hand to obscure part of their face.

This deep dive from Pragmatic Engineer offers details on how to quickly identify this type of behavior that should be included in any interviewer training for hiring managers conducting remote interviews.

2. Unauthorized assistance

Measuring a candidate's abilities is the reason technical interviewing exists. And like any form of assessment, some people will try to bend or break rules to give themselves an advantage. Whether it's seeking out commonly asked questions and copying answers from Leetcode, phoning a friend, or using an unauthorized AI tool, this type of candidate behavior will destroy a hiring signal if left unchecked.

Detecting and preventing this type of cheating comes down to monitoring specific behaviors that may indicate a candidate is getting outside help. For each of these examples, Karat has specific detection methods that Interview Engineers practice and learn.



Suspicious interview behaviors

1. **Continuous look aways:** The most common type of suspicious behavior (and the fastest rising) is for a candidate to frequently look offscreen or switch windows. Karat's Interview Engineers flag candidates who appear to reference another screen, device, or window while writing code or explaining their solution. This behavior may indicate the use of external tools or materials during the interview. Other behaviors that can indicate using external tools include highlighting or selecting the text of each coding problem, or reading each question prompt out loud, which are both methods of copying interview prompts into another tool via text or voice.

2. **Top-down coding:** The candidate produces the full solution in a single, uninterrupted pass, without adding conditionals, revisiting earlier logic, or making iterative changes. This type of behavior is uncommon in live problem solving and may suggest the use of pre-written or externally generated code. This is a perfect example of what differentiates live interviews from offline assignments. When you're watching someone problem-solve and generate code in real-time, it's easy to see if they are working through a problem or if they are copying down a memorized (or AI-generated) answer. Moving around a code base, renaming variables, running tests, and debugging along the way are inherently human behaviors that make it easier to identify AI usage in a live interview setting.

3. **Perfect answer:** The candidate immediately produces an optimized solution. In this scenario, the candidate jumps straight to a complete and optimal solution without first discussing their approach or working through the problem step-by-step. While this may reflect prior familiarity with the problem, it can also raise red flags when not accompanied by clear explanation. This is why probing into the "why" and "how" and not just the "what" is critical.

4. **Misaligned approach:** The candidate provides an incorrect or incoherent explanation. As in the previous example, the candidate writes a flawless solution line-by-line but gives an incorrect or illogical explanation of what the code does or how it works. This indicates memorization or copying AI-output without understanding. What's more, the "generative" nature of LLMs make it challenging for an overreliant candidate to produce consistent logic when asked a series of follow-up questions.

5. **Someone else heard:** The video recording captures a third party assisting the candidate. Interviewers are trained to flag any scenarios where someone else can be heard on the call providing answers, prompting, or coaching the candidate during the interview (note: you may need to consider the context of background voices if your candidate's name happens to be "Alexa" or "Siri").

6. **Matches online solution:** The candidate's code matches a known online solution exactly. The bottom line is there are only so many ways to solve some of the more basic algorithmic coding questions. This may suggest use of external resources but does not necessarily indicate misconduct, as many common solutions are open source or widely shared. This is why it's critical to have a robust and constantly evolving database of interview questions. Karat conducts regular searches on common answer-sharing websites, issues takedown requests when copyrighted interview questions are visible, and retires questions regularly to ensure overall interview integrity.

7. **No edge cases:** The candidate claims the solution is correct but is unable to name edge cases or explain how their code handles them. This behavior indicates a lack of understanding with respect to how the candidate arrived at their solution.

8. **Pasted solution:** The candidate pastes a complete block of code into the IDE without discussing their approach or developing the solution during the interview. This behavior may indicate the use of external tools to generate code. They may also paste partial solutions with inconsistent formatting, including the inconsistent use of spaces and tabs or inconsistent variable naming conventions.

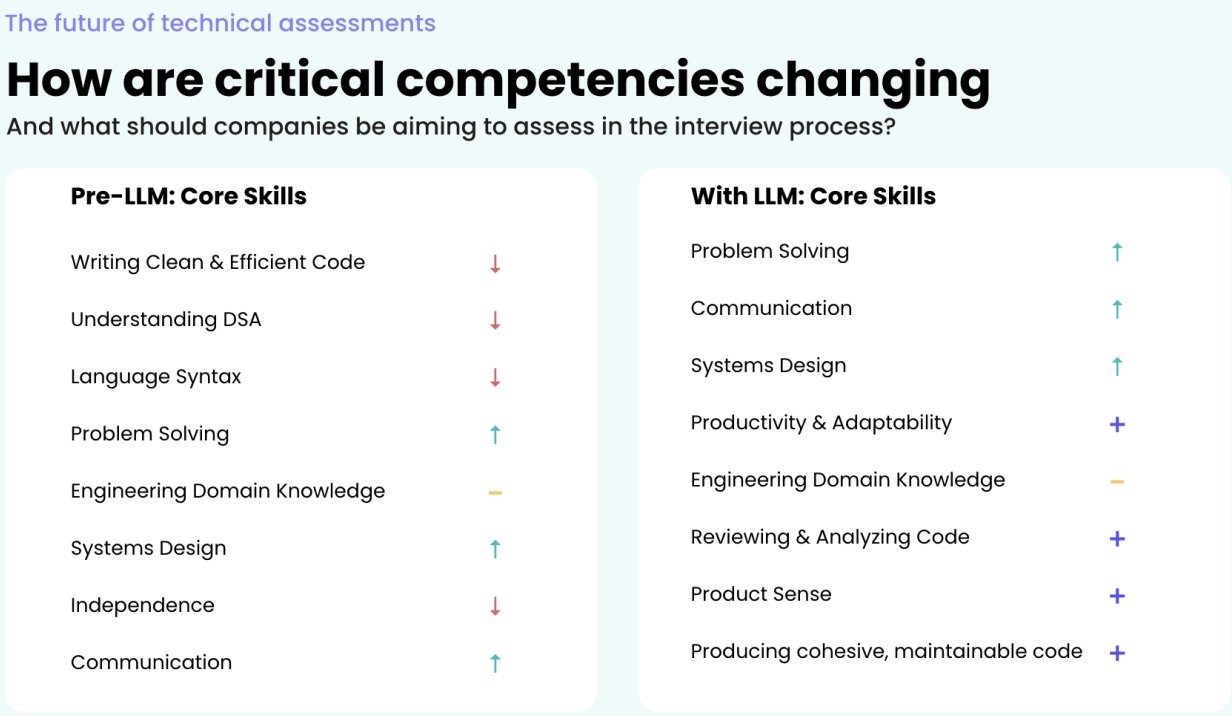
9. **Other Potentially Suspicious Behaviors:** Interviewers are coached to monitor and flag any unusual behavior not covered above that raises concerns, such as unexplained pauses, inconsistent logic, or sudden changes in communication style.

Building the AI-augmented interview of the future

The question isn't if AI should be enabled in interviews, it's when.

Meaningful interviews measure the competencies required on the job. As AI's role in software development grows, interviews will need to evolve. Today's requirements like memorizing syntax and data structures are irrelevant in a world where LLMs are producing meaningful amounts of code.

As this transformation occurs, interviews will need to focus on the underlying skills that make strong software engineers: e.g., problem solving, systems thinking, or handling edge cases. Engineers who possess these skills are going to be even more valuable in the future as AI turbocharges their productivity.



We're starting to see more companies that are growing comfortable with candidates using AI. This will become the norm as AI adoption in software development accelerates. We've always operated in an "open book" interview environment by allowing the use of Google or Stack Overflow. Today, the notion of prohibiting an accountant from using a calculator or spreadsheet in an interview seems preposterous. Similarly, allowing candidates to leverage AI is the next logical step for technical interviews.

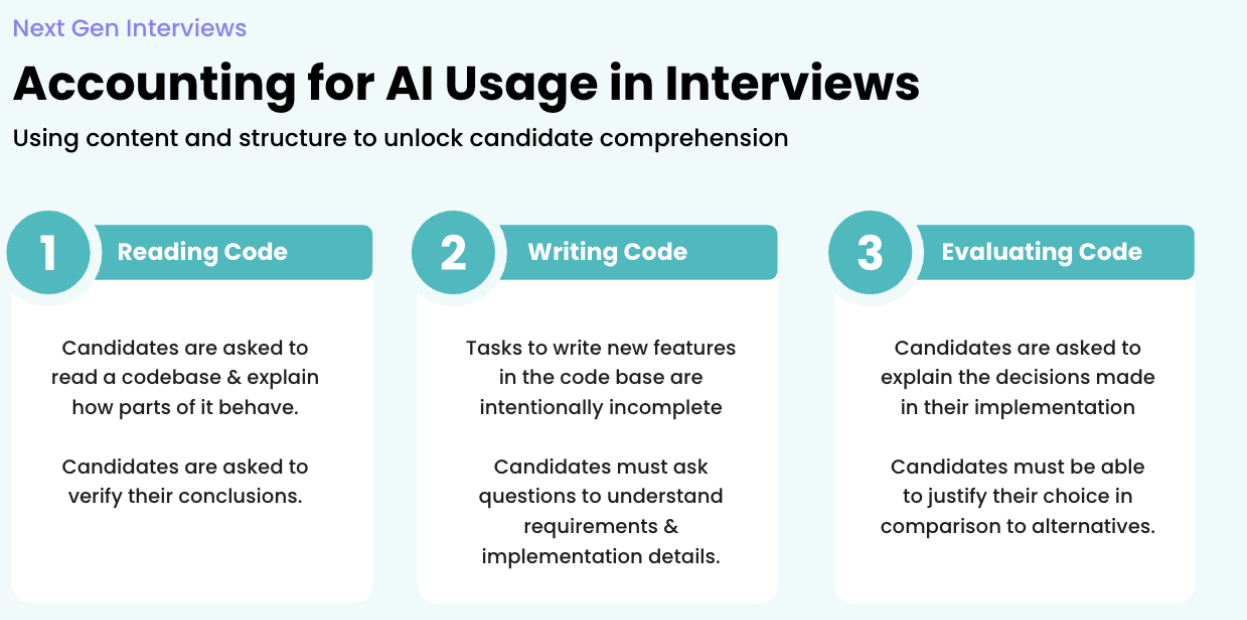
What's more, as AI-generated code continues to proliferate, the contextual awareness of engineers must grow. Because AI-generated code has limited context, it can introduce bugs, which is becoming a growing problem. Good engineers understand why and where bugs may occur, how systems are supposed to work. They are expected to apply these foundational concepts needed to thrive in a

high-quality engineering organization. The next iteration of interviews will continue to closely mirror a real-world set of problems for software engineers, rather than focusing on basic coding tasks.

All of this is why Karat is investing heavily in developing the next generation of interview formats that integrate LLMs while assessing the competencies that will matter most in the future. Our recent Byteboard acquisition is a key part of this strategy, enabling us to design assessments that align with the changing way that engineers work in the age of AI.

In this changing world, relying on output alone no longer produces a strong hiring signal. Having a live interviewer engage and probe candidates on the "why" behind their output, and their understanding of how systems work, allows a candidate to truly demonstrate the skills they need to perform on the job.

This new format uses a complex code base and AI-enabled IDE to produce a strong hiring signal. Candidates can ask the LLM coding assistant questions about the code base and use it to answer questions during the interview. In this environment, Interview Engineers will function more like a coworker than a test proctor. The interview results will measure a comprehensive set of skills including AI-literacy.



Conclusion

Effectively raising your hiring bar requires a more strategic approach than seeing whether or not a candidate can solve a coding problem that your interviewer found on a deep corner of the internet. The complexities of AI-enabled software development are forcing organizations to overhaul their entire hiring process.

Interviews need to be far more complex. The infrastructure to support such interviews (including multi-file codebases and LLM-supported IDEs) requires far more work than spinning up the algorithmic interviews of the past. Interview content needs to evolve beyond reversing a string, and interviewers need to be more adept at engaging candidates to produce a reliable hiring signal.

But this is just the beginning. AI tools will continue to evolve at a rapid pace. And measuring talent still requires a deliberate approach and methodology, even as that methodology evolves.

Interviewing is no longer a transactional task. Maintaining a relevant and effective interview practice is a full time discipline. It's a strategic competency where consistency and accuracy remain the foundation. And the organizations that treat interviewing strategically will win the race for talent.