Podcast Summary: Why Top Engineers Are Moving Beyond Output-Based Coding

The Signal in the Noise

The engineering podcast ecosystem has become the watercooler of the industry. When you listen closely to what engineering leaders are saying week after week, patterns emerge. And right now, one pattern is impossible to ignore: **the industry is moving beyond output-based coding tests.**

This isn't a fringe opinion. It's a consensus forming across the most respected voices in software engineering—leaders who have conducted thousands of interviews, CTOs who have scaled teams from startups to enterprises, and engineers who have seen what actually predicts success.

Here's what they're saying—and why it matters for your hiring process.

Why Multiple Coding Rounds Don't Give More Signal

Ashwin Baskaran, VP of Engineering at Mercury, has been in engineering leadership for over 20 years. In his YouTube podcast *High Output: The Future of Engineering*, Baskaran made a point that cuts through the noise [1]:

"I would say interviews in the early part of when I became a manager tended to be very technical. Multiple coding rounds. But simply doing more technical interviews doesn't necessarily give you more signal."

His observation is backed by experience: a single well-designed coding interview gives you a more meaningful signal than four or five ad hoc brain teasers. The companies still doing marathon coding sessions are missing the point entirely [2].

What Actually Predicts Success

Baskaran argues that the real evolution in hiring is recognizing that engineers need **product sense**—not just technical ability [1]:

"I think the typical product company has more of an expectation that engineers will have ideas on the product and have product sense and not outsource their thinking."

At Mercury, Baskaran literally asks candidates: **"Define the product."**

He leaves it vague on purpose. Whether it's infrastructure for databases or tools for general contractors, he's looking for people who understand boundaries—who built this, who experiences it, how they're experiencing it [1]:

"I'm looking for people who have the sense like, there's a boundary and this boundary is being experienced by somebody."

This is a fundamentally different signal than "can you write a sorting algorithm on a whiteboard." It measures judgment, context awareness, and the ability to think beyond the code.

The AI Testing Ground

Baskaran's team at Mercury has been experimenting with AI-augmented development across their organization of over 250 engineers. Their hypothesis? **AI will be a net positive—but it's going to be a journey** [1].

The key insight is that different problems benefit differently from AI assistance:

- **Greenfield applications:** AI shines here. It can generate code quickly and effectively.
- **Legacy systems with complex context:** Much harder. AI struggles to understand all the dependencies and constraints.

This creates a new interview challenge:

"Do we change our interview structure and measure for proficiency with AI tools?"

Mercury is exploring screening processes that actually watch candidates use coding assistants in real-time. Not just prompting—sophisticated use of the tools integrated into their development environment [1].

The question isn't whether someone can code, but how effectively they can collaborate with AI.

Vibe Coding Is Not a Career

The *Develop Yourself* podcast, hosted by Brian Jenney on YouTube, dedicated a full episode to "8 Harsh Truths for Software Developers in the AI Era." The first truth is a warning to every hiring manager [3]:

"Vibe coding is not a career or a replacement for actual skill. If you're copying and pasting stuff from ChatGPT and passing it off as your own work, that's not a career. It's not even really a skill."

The host shared a personal experience [3]:

"I interviewed some people last year. They were basically caught cheating. They were vibe coding in front of us and they couldn't explain the code at all. We actually told them they could use AI during the interview. One of them said he was not. The other one said he was a software developer that had used the languages and frameworks we were testing for. But it was super clear in both interviews that neither one could probably do a for loop on their own."

This is the new reality of technical hiring. Candidates who rely entirely on AI-generated code are hitting a wall when things get complicated. They can produce, but they can't debug. They can ship fast, but they can't maintain [4].

The Shift Beyond Algorithmic Puzzles

The *Beyond Cracking the Coding Interview* podcast features Mike Mroczka, an interview coach and former Googler, discussing what really goes on behind technical hiring [5].

Key insights from the episode:

- How leaked question banks and standardized puzzles can distort hiring signals
- Practical ways companies can make interviews fairer and harder to game
- A balanced take on data structures and algorithms: when they're useful and when they're noise
- How to structure interviews for different seniority levels so you measure the right skills

The episode addresses a crucial point: many "perfect" candidates simply learned the test—while great engineers get filtered out by bad interview design [5].

The Architecture Advantage

Stephen Houston, CTO of TeamFeePay, made a similar point. Despite his enthusiasm for AI, he emphasizes what doesn't change [6]:

"The principles of software development, software engineering have never really changed. Good architecture, good designs, following design patterns, thinking through what it is you're actually going to do before you actually do it—they haven't changed. My software design pattern book is now over 40 years old. The patterns are still the same, they're still valid today."

His prediction for what will separate top performers:

"The real skill is judgement: knowing when an answer is plausible but wrong, catching shortcuts like hard-coded outputs, and encoding lessons into prompts so the whole team compounds."

In a world where AI can generate code, the premium is on engineers who understand what to build and how it fits into the larger context.

What This Means for Your Hiring

Across these podcasts, a consistent framework emerges for modern hiring:

1. Move Beyond Coding Marathons

Multiple coding rounds don't give you more signal. One well-designed interview does more. Focus on quality, not quantity.

2. Assess Product Sense

Ask candidates to define product boundaries. See if they understand who builds what and who experiences it. This reveals judgment and context awareness.

3. Test AI Collaboration

Watch candidates use AI tools in real-time. Evaluate how they prompt, refine, and evaluate output. The skill isn't just generating code—it's knowing what to do with it.

4. Screen for Judgment, Not Just Implementation

In a world where AI can generate code, the premium is on engineers who understand what to build and how it fits into the larger context.

5. Look Beyond Standardized Puzzles

Design interviews that are harder to game and that measure the right skills for each seniority level.

Time to Rethink Hiring

The leaders who are evolving their hiring practices now are building significantly stronger teams than those still stuck in output-based assessments.

The question isn't whether your process needs to change—it's whether you'll change before you lose the talent you need to compete.

The evidence from the engineering podcast ecosystem is clear: the future belongs to engineers who combine technical depth with product sense, AI collaboration skills, and sound judgment. And the future belongs to companies that know how to find them.

So ask yourself: When was the last time you audited your hiring process? When was the last time you asked whether your interviews are measuring what actually matters?

Because the engineers you're trying to hire are watching these conversations. They're already moving forward.

The question is: will you move with them?

References

1. https://www.youtube.com/watch?v=tl2TIhq9Sao
2. https://getmaestro.ai/podcast
3. https://www.youtube.com/watch?v=CmaRHRMslB0
4. https://www.podscan.fm/player/develop-yourself/303-8-harsh-truths-for-software-developers-in-the-ai-era
5. https://www.youtube.com/watch?v=SNf8TwRfTtg
6. https://www.listennotes.com/de/podcasts/the-digital/why-developers-who-only-code-7t0FsdyDlQf/?__cf_chl_f_tk=TW1Ye_gV4i64pxASTjlzsN8TlIi5eShyUMB3sXfUY.o-1783442449-1.0.1.1-ieC.C6y3mSjgyxMuxG1SFb35yro4VdXV4BiqkxGpwt8#google_vignette