Zero-Shot Learning - Episode 8: Rohan Varma from OpenAI

Code is the easy part

Hosts: Nancy Wang, Chief Technology Officer, 1Password Dev Tagare, Senior Director and Head of Engineering, Gemini Enterprise and Business, Google (personal capacity) Guest: Rohan Varma, Codex, OpenAI

---

1. Introduction

Nancy Wang: Hello everybody, and welcome to Zero-Shot Learning, a podcast about the reality of developing with AI from the people actually doing the work. I'm Nancy Wang, Chief Technology Officer at 1Password, and I'm joined today by my co-host Dev Tagare, Senior Director and Head of Engineering for Gemini Enterprise and Business at Google. Today we have a very exciting guest - Rohan Varma, who works on Codex at OpenAI and who was previously at Cursor. Rohan, welcome to the show.

Rohan Varma: Thanks so much for having me. Really excited to be here.

Nancy Wang: So let's start with the obvious question - you went from Cursor to OpenAI to work on Codex. Two very different companies, two very different products in the same broad space. What actually changed when you made that move?

---

2. From Cursor to Codex: What Changes When You Move to the Cloud

Rohan Varma: Yeah, they both operate in very different ways. I think what's really interesting about Codex is that it's fundamentally a cloud-based agent. The model of Cursor is very much like - you're there, you're in the IDE, you're interacting with the agent directly. The model of Codex is much more asynchronous. You send it a task, it goes and does the work, and then it comes back to you. And I think that shift is actually really profound because it changes how you think about what AI can do for you.

The other big difference is just the scale of what Codex can do. Because it has access to a computer - it can run code, it can use a browser, it can do things that an IDE-based assistant fundamentally can't do. So it's able to take on longer-horizon tasks that you just couldn't run in the background while you were doing other things.

Nancy Wang: And tell us a little about the team itself - how you think about building Codex at OpenAI.

Rohan Varma: Yeah, so the Codex team is actually pretty small. We have something like two PMs, two designers, and then somewhere between 30 and 40 engineers floating across probably 15 to 20 different product areas. And I think that's actually kind of representative of where things are going - you have a relatively small, highly leveraged team that's able to move really quickly across a lot of different surface areas.

And I think like that's actually been a really interesting learning for me, which is that the team structure itself has to change when you're building with these tools. It's not just that the same team moves faster - it's that the optimal team looks different.

---

3. The Adoption Gap: Why P99 Users Are 100x More Leveraged Than P50

Nancy Wang: So let's talk about adoption. Because I think there's a real gap between what people think these tools can do and what they're actually getting out of them. What do you see in terms of how different teams are using Codex?

Rohan Varma: Yeah, this is something I think about a lot. And I think the honest answer is that there's a massive variance in how much people get out of it. I think the P99 Codex user - the person who's really getting the most out of it - is probably 100x more leveraged than the P50 user. And the P50 user is still maybe getting like 20% faster or something.

And the difference isn't really about the tool. The tool is the same. It's about how they're using it. The high adopters treat Codex like a team member, not like a smarter autocomplete. They give it context. They specify outcomes clearly. They let it figure out the approach. And critically, they're not sitting there watching it work - they send it off, they go do something else, they come back to the result.

The low adopters are using it more like a really good search engine or a code suggestion tool. They're getting incremental improvements, not transformational ones. And I think part of what we're working on is helping people make that transition - because the jump from P50 to P99 is actually achievable, it's just not obvious how to get there.

Dev Tagare: Yeah. Like whoa.

Rohan Varma: Right? And I think what's interesting is that it's not really a skills gap in the traditional sense. It's more of a mindset shift. You have to be willing to give up control of the how and focus on the what and the why. And that's actually a harder thing to do than learning a new tool.

---

4. The Real Bottleneck: CI/CD and the SDLC After Code

Nancy Wang: So let's get into something that I think is actually the most interesting part of this conversation - which is the bottleneck. Because I hear a lot about "AI makes you write code faster." But what you're describing sounds like writing the code is actually not where the time is going anymore.

Rohan Varma: Yeah, exactly. And I think this is like the key insight that I think gets underappreciated. Like, writing code is the easy part. Like, literally, that is like the easiest thing. Like, Codex can write code really fast. The hard parts are everything that happens around the code.

Like, okay, so let's say Codex writes something in like five minutes. What happens next? You have to run CI, which might take 20 minutes. You have to get a code review, which might take hours or days depending on your team. You have to get it deployed. You have to verify it worked. All of these things are now the bottleneck. And the challenge is that our processes - our CI systems, our review processes, our deployment pipelines - were all designed around the assumption that writing the code was the slow part. And that assumption is just no longer true.

Nancy Wang: So what do you actually do about CI? Because I feel like a lot of companies can't just flip a switch and have faster CI.

Rohan Varma: Yeah, I mean, it's a real problem. And I think some of it is just - you have to prioritize it in a way that maybe you didn't before. Like, historically, if your CI takes 20 minutes and your engineers are writing code for hours before they submit, that's fine. The CI is fast relative to the development time. But when Codex is submitting ten things in an hour, suddenly 20-minute CI is a massive bottleneck.

And so I think what it means is that CI infrastructure becomes actually much higher leverage than it's been historically. And things like parallelizing CI, like caching, like being much more smart about what you actually need to run - those things matter a lot more than they used to.

But also I think the deeper thing is just that the whole SDLC needs to be rethought. Like, it's not just CI. It's like - what does code review look like? What does deployment look like? What does verification look like? All of those things need to catch up to where code generation is.

---

5. What the High-Leverage Team Actually Looks Like

Nancy Wang: Let's talk about team structure then, because I think this is where your perspective is actually really unique - you're inside OpenAI watching this happen, you're seeing the Codex team itself operate this way. What does the team that's really getting leverage actually look like?

Rohan Varma: Yeah, I think the biggest thing is that the roles start to blur. Like, historically you had - engineers write code, PMs define what to build, designers figure out how it looks. And those were pretty distinct. And I think what's happening is that the engineers are being pushed up the stack. They're doing more of the product thinking, more of the system design, more of the "what should we build and why" - because the actual implementation work is increasingly handled by the agents.

And so the optimal team - at least what I'm seeing - is actually smaller but with a different composition. You don't need as many people whose job is purely writing code. You need people who are really good at specifying what needs to be built, at verifying that it was built correctly, and at setting up the systems that let agents work effectively. Those are actually pretty different skills.

And you know, our own team - the Codex team - is kind of a living example of this. A small group of people floating across a huge number of product areas because the agents are doing a lot of the implementation work. Every engineer is doing more product work than they would have done historically.

Nancy Wang: And is that a harder job or an easier job?

Rohan Varma: I think it's a different job. And I think it's one that a lot of engineers actually find more interesting. Because now you're spending more time on the things that actually matter - like, what should we build? Does this actually work? Did we get the outcome we wanted? - and less time on the mechanical parts of translating a spec into code.

But I also think it requires a different kind of discipline. You have to be really good at specifying things clearly. And that turns out to be a hard skill that a lot of engineers haven't had to develop because historically the specification was the PM's job.

---

6. All Code Is Vibe Coded Now - So Do It Well

Nancy Wang: I want to ask you about vibe coding specifically, because I feel like there's a lot of confusion about what that term actually means at this point. Like, where does it end?

Rohan Varma: Yeah. I think honestly, like, all code is at this point quote unquote vibe coded. Like, everyone is using AI to write code. The question is just whether you're doing it well or not. Like, vibe coding in the pejorative sense is like - you just kind of paste something in, you don't really understand what it does, you ship it and hope for the best. And that's obviously bad.

But the alternative isn't to go back to writing every line yourself. The alternative is to be really intentional about how you use the agents - to specify clearly, to verify rigorously, to understand the output well enough to know if it's right. And that's actually a much higher bar than just writing the code yourself in some ways. Because you have to understand both what you want and be able to evaluate whether you got it.

So I think the framing I'd push back on is this idea that vibe coding is some new category. Like, every engineer using Cursor or Codex is vibe coding. The question is just whether they're doing it with rigor or without it.

---

7. Agentic Code Review and Prompt Provenance

Nancy Wang: Let's talk about code review, because I think this is where things get really interesting. If Codex is writing the code, what does review actually mean? What are you reviewing?

Rohan Varma: Yeah, I think this is a place where the whole framing needs to shift. Because historically, the thing you were reviewing was the code. You were looking at the diff and asking - does this make sense? Is this correct? Is this the right approach? And that made sense when an engineer wrote it, because the code was evidence of the engineer's reasoning.

But when an agent wrote it, the code is kind of beside the point. Like, the agent can write correct-looking code that does the wrong thing really easily. So reviewing the code itself doesn't give you that much signal. What you actually want to review is the prompt. What was asked for? What was the intent? And then verify - did the output match the intent? Because that's actually where the human judgment needs to go.

And then I think the ideal world is actually that the verification step itself is automated. Like, the agentic review ideally is actually more of like a verification step. And if it can happen fully verified, then that should just happen when the code is generated. So Codex comes back and it's like - here's a PR, it's been fully validated, it's been reviewed by other agents. Here's me clicking the demo to prototype. And so now you're just like - I'm ready to ship.

Nancy Wang: Yeah. I mean, my ideal would be - you repro the issue, and then you're actually able to generate the PR and actually show it working in production.

Rohan Varma: Yeah. And that's a great example where Codex is probably going to be way more persistent about going and finding all the possible evidence - giving you that writeup, making the code change, and then proving it to you. Like, let me remove the code, show the issue, add the code back, show the fix. I think we'll get to that level of verification with agents much more - to high fidelity - than we would have without agents.

Nancy Wang: How soon do you think that will be?

Rohan Varma: I'd say we're doing it. It's largely a function of the tools that the agent has to run with. Today, Codex can definitely do things like record demos, run tests - a lot of that stuff - because it has access to computer tools, it can already do it. It's kind of like - what is the process that you want it to do? And then actually codifying that process with Codex.

---

8. When Codex Talks to Codex

Nancy Wang: Let's talk about what happens when agents are interacting with other agents. Because I think that's the next thing that people aren't really thinking about yet.

Rohan Varma: Yeah, I think this is like really underappreciated. The thing I'm most excited about is the idea that a lot of the communication overhead that exists in organizations just kind of goes away. Like, so much of what we do within an organization that has more than one person in it is just moving information around.

And I did this experiment the other day where I needed to collect feedback from our account directors on a specific feature. And I had Codex go and ask this question identically to all of them. And some of the account directors responded with Codex as well - their Codex was responding to my Codex. And my Codex was updating the spreadsheet. So it's like -

Nancy Wang: My agent talks to -

Rohan Varma: Your agent. Yeah. And I think that's actually great. Because now the information is being maximally spread out. It's happening automatically. Humans are getting the information when they need it. Like, already, typically any time I have a question about anything, I just ask Codex to tell me about it and it'll give me the most up to date information from whatever Slack channel I'm not even a part of. And so that's just fundamentally a better flow of information than, you know, outdated docs that aren't getting updated.

---

9. What Gets Obsolete in 12 to 24 Months

Nancy Wang: So looking into the future - let's fast forward 12 to 24 months, which is a long time in today's world - what part of the developer workflow do you think gets obsolete or just gets completely replaced in the agent environment?

Rohan Varma: Like 24 months from now? We'll see what happens.

Nancy Wang: Maybe we'll be sitting on the beach.

Rohan Varma: Yeah, yeah. I think that probably, if we're able to accomplish what we're trying to accomplish, engineering does fundamentally look different. I will say there will be a long delay of work required to actually deploy these benefits to the vast majority of engineers and enterprises in the world. So maybe the most frontier companies are going to be operating in this extremely unrecognizable way, but the majority of companies will still be adopting slowly. We'll be doing the work at OpenAI to help them do that.

But I think fundamentally, engineers are just moving into more and more leveraged positions. As an engineer, you're no longer writing and reading code. You're working with a bunch of agents, defining systems, creating these verification loops, these meta-harnesses to have agents run more effectively on important tasks, and creating these self-improving systems. We're already doing this at OpenAI. There's an engineer on our team, Ryan, who wrote an article about how we built a self-improving harness for a product that was shipped fully and authentically. And I think we're just going to see more and more of that, and probably less and less of traditional software engineering.

Obviously that's going to hit different types of engineering, different types of roles, different types of companies more or less gradually. But I think that's probably the ideal state - because now every person is way more leveraged, doing way more productive work, accomplishing more. Companies can just do more. So hopefully more and more software will get built that's more useful for people.

Nancy Wang: So do you think the IDE will still exist as we know it?

Rohan Varma: Oh, for sure not. I mean, I think fundamentally the IDE is already shifting. There will be a long tail of adopting the future, but the Codex app is maybe like an intermediate step on the journey towards long-running virtual coworkers and full cloud agent capabilities.

The thing that my head goes to is that a lot of the interfaces we built to work with humans are probably similarly correct interfaces to work with agents. And I think the question is just - is the agent you're working with like an IC? Or is the agent you're working with like a director of engineering with 60 engineers under them? And under the hood, they're just delegating to a bunch of sub-agents. But over time, it'll just feel like you have more and more collaborators who are able to do more and more work.

---

10. The Habits Worth Keeping and the Ones to Drop

Nancy Wang: And what's one habit you hope survives the move to agents? And what's one habit you hope disappears?

Rohan Varma: I think the habit I hope disappears is basically manual work to spread information. So much of what we do within an organization that has more than one person in it is just moving information around. And literally any communication - even internally between customers, between clients, between vendors - all that communication can just be done automatically. I think we're starting to get to the point where we can programmatically route information to the right people at the right time.

Something that I think will probably need to stay - which is actually ironic given what I just said - is writing. I hope writing is still the thing we do. Because I think well-written communication is actually extremely useful to communicate clarity of thought. Which is kind of contradictory to what I just said I hope goes away, but - I think that a lot of what our jobs are going to be as humans is fundamentally deciding what needs to get done, agreeing upon what needs to be done, and being really good at aligning ourselves and the agents working on it. And so well-written documentation of an idea, and clarity, is something that's going to be still very important.

Nancy Wang: Yeah, you'd be surprised. I think especially now with everybody using LLMs for writing docs, you can quickly see what's AI-generated and what's not, or at least what's the thought behind the prompt.

Rohan Varma: Totally. Yeah. I think when it's directly an agent - at least when you use Codex in Slack, it'll actually say "sent by ChatGPT." But yeah, you can tell. And it's funny, because when it is really long, you almost want to say: what was the prompt that was used to generate this? Just send me the prompt next time. And it's kind of like - if your favorite example is someone one-shots a PRD with ChatGPT, and then someone puts that into another LLM and asks it to summarize, it's just... okay.

Nancy Wang: Exactly. If your favorite example is someone one-shots a PRD with ChatGPT, and then puts that into another LLM to summarize - it's just summarizing back to nothing.

Rohan Varma: Yeah. I feel like I think of this like with recruiting, where someone generates their application and their resume with AI, and then an AI system on the company side is reviewing it. And at a certain level, we should just have some efficiency of the AI talking to each other directly. So I think it's definitely an interesting time to be working in a company right now.

---

11. Closing: What Would You Build?

Nancy Wang: Alright, one final question - we always ask this of our guests. If you could build anything in the world - and it could actually be physical as well - not necessarily Codex, though it could be - what would you build? Let's say you were taking a six-month sabbatical.

Rohan Varma: I'd probably build an AI tutor or something like that. I think in general, what I'm really excited about - and this is part of why I'm excited about working at OpenAI - is that this technology fundamentally makes things that were previously inaccessible super accessible. And I think there's a concept that a friend of mine who's an investor shared, which is that good consumer technology just makes inaccessible things accessible. Like private chauffeurs - now you have Uber. Instant communication - now you have video chat. And I think there are all these things that are blocked on needing a human to do it, but there aren't enough of those humans. In education, for example, we know that the best way for someone to learn is one-on-one tutoring. Across every form of learning. But that's fundamentally inaccessible to the majority of people in the world. And I think that's something that has to happen. Will probably happen. I probably won't be the one that does it, but someone should. And there are a lot of really smart people I know working on it.

Nancy Wang: I thought that was going to be your answer. And we could probably take this for another couple of hours. One of the things that I was super excited about with online learning is the ability to create courses. We've actually published a bunch of courses on Coursera, including one on AI product development, that folks can access for relatively very little cost. But I

feel like especially with ChatGPT and LLMs, it's really changed learning as well. People want on-demand answers. I'm curious - what role is ChatGPT or LLMs in general playing with education?

Rohan Varma: Yeah, it's super evolving and I think it's changing a lot of what's important to learn as well. And at OpenAI, in pursuit of our mission - which is to develop AGI and distribute it usefully and safely to the world - I think it's definitely our responsibility to help push a lot of this forward and figure out what the world looks like when it's different. What is important to know, what is important to learn, what's important, and all of that. So yeah, a lot of interesting change and a lot of exciting stuff to think about there.

Nancy Wang: Yeah, perhaps a topic for our next episode. Well, thanks so much for coming in the studio today.

Rohan Varma: Yeah, thanks for having me.
