Zero-Shot Learning - Episode 5: Adarsh Hiremath from Mercor

Evals are agent infrastructure

Hosts: Nancy Wang, Chief Technology Officer, 1Password Dev Tagare, Senior Director and Head of Engineering, Gemini Enterprise and Business, Google (personal capacity) Guest: Adarsh Hiremath, Co-Founder and Co-CEO, Mercor

---

Nancy Wang: Hi and welcome to the latest episode of the Zero-Shot Learning podcast, where we talk to leading AI builders about what's actually happening as they're building with AI. My name is Nancy Wang, and I'm the Chief Technology Officer at 1Password.

Dev Tagare: And I'm Dev Tagare. I am the Senior Director and Head of Engineering for Gemini Enterprise and Gemini Business at Google. The views here are my personal views - that's a legal disclaimer I always have to say anytime I talk about AI.

Nancy Wang: But it does get pretty spicy.

Dev Tagare: It does get spicy. Always. Welcome. Lovely to have you on the pod.

Adarsh Hiremath: Thanks for having me.

Nancy Wang: We talk a lot about how Mercor has been helping enterprises and businesses capture the way they do work. Well, what does good work actually look like inside an organization? And secondly, how can you actually evaluate whether AI systems are performing real work - potentially in the world of software engineering - versus just narrowly prompting or helping them ace coding tests? As we dig into the technical side of that loop, let's talk about how you at Mercor build reliable human workflows using agents.

---

1. How Mercor Works: Discover, Deploy, Improve

Adarsh Hiremath: So for the past couple of years, we've spent a bunch of time supporting the labs with training, deploying, and evaluating agents - training meaning making them better, deploying meaning making them practical, and evaluating meaning the evals that help us understand whether or not the agent is behaving as we intend it to.

Very quickly, we realized that a bunch of enterprises started reaching out to us asking for help with the exact same thing. We want to train agents, we want to deploy agents, we want to evaluate them. And Mercor became a preferred vendor for that. So we support everything from hyperscalers who want to build evals for every single one of their departments, to companies that have a specific use case and a specific agent they want to build.

The two challenges we've observed come from telling the agent what to do - thing one - and then evaluating whether or not it's doing it well - thing two. In an enterprise context, telling the agent what to do has its own set of complexities. The most valuable big-rock problems to solve often have context that's like tribal knowledge deep within an enterprise. And even if that knowledge exists, it's not in a digital space that's actionable or useful for an agent. So actually discovering that content, pulling it out, and putting it in the specification of an agent is a really challenging problem. Then the second thing is, once that agent is live and in production, how do you understand whether or not it's performing well?

This is where Mercor's bread and butter comes in, which is evals. You need to understand what the human workflow looks like, what success looks like when the agent attempts that workflow, and when it fails, how do you have a continuous learning loop to make sure that mistake never happens again? We've productized that whole thing and are now working with enterprises on their agent deployments to make sure they're as reliable as possible.

Nancy Wang: So given that the problem space for many businesses is so vast - take 1Password, for example, where we're looking to build dev-specific workflows, while another company might want to build customer support or recruiting-specific workflows - how do you even know what good looks like in that case?

Adarsh Hiremath: Yeah. Even just taking the example of 1Password - the key thing is it's not just an agent that does dev workflows. It's an agent that does dev workflows within 1Password. For that agent to do a good job, it needs to know what 1Password considers good for a specific role or a set of developer tasks. Even for simple workflows, the general intelligence of the model is not sufficient. Let's just say reviewing a resume - you could take the smartest model and have it review your resume, but if it's not calibrated with your recruiting team and what that team is looking for, it's just going to fall flat on its face.

So the question of how we understand what good looks like requires extracting knowledge from the enterprise. That can be everything from connecting to SaaS tools, uploading documents, conducting moderated interviews of people actually doing the job. And the second thing is an eval that helps us understand the failure modes. Specifically, the eval covers the task the model is trying to do, the trajectory or reasoning steps the agent is taking, and then the actual outcome graded against a set of subjective or objective criteria - what we call a rubric.

---

2. Signal vs. Noise: From Quantitative to Qualitative Evals

Dev Tagare: Makes sense. Just double-clicking into this a little bit - in any enterprise context there's always this thing of expert judgment. Enterprises are deep in a specific domain area. When you go from quantitative data to expert opinion or expert judgment - more qualitative, more intuition-driven - there's a lot of spectrum for signal versus noise in that journey. One, how do you solve for it? And two, when you really dig into this, are there any failure patterns or modes that have surprised you more than others?

Adarsh Hiremath: Addressing the point about signal to noise - that is very, very use-case specific. For example, in customer support there are really only two metrics you care a lot about. One is: did the AI resolve the ticket end to end - that's ticket resolution. And two: did it have a customer satisfaction rate better than or on par with a human? Those are two things that are very, very easy to hill-climb with the right tools, and they're quantitative as opposed to qualitative.

There are other use cases where the value judgments are entirely vibes-based. And that's a really, really important thing to get right. For example, if you're trying to automate a consulting workflow that produces a slide deck, it's not cut and dry like customer support resolution. The question becomes how do you structure the preferences of the person who knows what good looks like for that specific task into a rubric format. You can break it apart into the discrete things that you'd look for if you were a human evaluating the slide deck. An objective thing might be whether or not the slide deck sticks to a specific color format. But a subjective thing might be whether or not the actual content is not verbose, easy to understand, and follows a typical structure. So the key is combining both of those for tasks that are unbounded and subjective.

Dev Tagare: And in your journey, have you noticed some metrics that may look great on paper, but then in production when you see real-world use cases, they don't look as good? One thing I've noticed over time is that in search, you get answers quickly and everyone personalizes to some degree. But over time your preferences are distinct from mine, distinct from Nancy's. In search that was never a problem because you'd just get the ten blue links, ranked across many people. But are you noticing that individual user preference is now dominating even enterprise outcomes and gradations there?

---

3. Why Trajectories Matter More Than Outputs

Adarsh Hiremath: Yeah. One really important thing is understanding the inputs into an output, or the reasoning that actually leads to a specific output. Because you could have a model that just answers correctly the first time, but it's making all the wrong decisions along the way. Then when you adapt it to a slightly different context, all of a sudden you've got an agent that's totally broken in production. So the evals need to cover not only the task and the output - the rubric I just described - but also the trajectory that connects the two. You could have an LLM judge or a human evaluating whether the reasoning is sound. If the reasoning is sound and the output is achieved properly, odds are the overall deployment is going to go well - assuming you have a reasonable distribution of all three things: inputs, trajectories, and outputs.

To your point about whether output metrics can be deceiving - the answer is 100% yes. It's no different from metrics for a product team, where if you really wanted to game a metric, most product teams could. So it's about having the right guardrails. Going back to the customer support example - you could have a high resolution rate, but the way that the agent actually communicates with the human could be totally broken. Super rambly, verbose, including information that -

Nancy Wang: - argues everyone into a free flight ticket, you know?

Adarsh Hiremath: Or it could give everyone a free flight ticket, and there could be a glitch that pays the customer out $1,000 for every interaction. So you need quality control and evals over the whole chain of processes to understand whether it's actually doing the job well.

Dev Tagare: Along the same trajectory - how do you account for disaggregated learning? A feedback loop in the customer support example could be that my resolution rate is 80%, but then six months later I'm getting complaints on a specific arc or trajectory, and that's feeding back into the overall process. How does that work?

---

4. Continuous Learning and Hallucination Prevention

Adarsh Hiremath: Yeah. Maybe a high-level observation is that use cases where the agent has to balance a bunch of different stakeholder perspectives and aggregate a bunch of different contexts are the big-rock problems we want to solve. They're super hard because it's not as straightforward as just prompting Claude, Gemini, or whatever your favorite model is - because you're always there to steer it, you have a certain set of preferences. So category A of problems with many stakeholders and a bunch of different context sources are the really tricky ones to solve.

The way we've gone about it is making sure you can actually index the entire company's context by pulling in all the different sources. That context is typically beyond a single person or a single data source - it's supposed to be a comprehensive and holistic view of what good looks like for that company. And then the second thing is what happens when there's actually a failure. Models can actually identify these failures - if it's not confident in an answer, it can escalate and triage to a human. Typically the first round of learning updates we want to do are in context space, not parameter space, because it's way more compute-efficient. You triage it to a human, update the skill. And if it warrants it, you can do fine tuning on the model if you're using an open-source model. But given how smart the models are, the former is actually usually the better approach.

Nancy Wang: Yeah. So coming to the technical deep dive - you've started to unpack the layers. Going back to your point around how work is actually being done, and not just handcrafting prompts and sending them out with best hopes - what are some platform primitives you're building to contextualize all of that information?

Adarsh Hiremath: The platform we use has three components: discover, deploy, and improve. Discover is where you connect with all the data sources of a company - SaaS applications, the actual subject matter experts who know what good looks like, documents - you're able to pull it out and put it in a digital space that's accessible and retrievable for an agent, contextualized to that company. The second thing is actually deploying the agent. Humans typically interact with these agents in platforms like Slack - that's where work happens. You want to invoke the agent where you're already doing your workflows, and you want to make the UX such that people are excited to use it, not pained by it. And then the last is the improve step, which is where a lot of the magic is. You want an eval set to figure out where the models are failing. When you do find a failure, you want to one, update that eval set to make sure that failure never happens again, and two, take the piece of context that addresses that model failure, suggest a diff to the skills file, and put that in the specification of the agent.

Dev Tagare: Makes a lot of sense. So to paraphrase - the updates are happening via skills. Skills are learned by a combination of human interactions across problems and surfaces. You triangulate those with other evidence and grounding data you collect over time to continuously update skills and push that back as your feedback loop into the system.

Adarsh Hiremath: Exactly.

Nancy Wang: So take an example that crosses many different functional groups - like a demand gen pipeline. That crosses marketing, sales, and can also touch product teams because it might talk about certain feature requests. Where do you draw the line between superstructure and specification versus tacit knowledge that is very contextualized to a specific company?

Adarsh Hiremath: The question is where you draw the line between company-specific and skills that are general. Thankfully there's a lot of overlap. A model with really good underlying reasoning capabilities will likely do a better job of demand gen than a model with bad reasoning - but the part that's really cool about a use case like that is that it involves so many different departments and context sources. Just to give one example of a deployment for that use case: you'd want the agent to actually have access to all the different releases that are happening - something like Jira, a Slack channel with releases. Second, it would want the typical guidelines of a marketing department - what is the actual piece you'd put out to get demand, an email outbound designed to get demand. That could be Google Docs, emails, onboarding documents for the marketing team. And third, you'd want to give it the ability to do the outbound motion - whether that be email, socials, or other things. And let's say you queue up a piece of material that is not desirable, or one of those stakeholders flags it as incorrect - you can go into the platform, update it, suggest the diff, and so on over and over again for a bunch of different use cases.

---

5. The Ontology Problem: Semantic Meaning Across Companies

Dev Tagare: Makes a lot of sense. So we're sort of entering the context discussion. One thing that keeps coming up, Adarsh, is the ontology for context - different companies call similar things differently. I may call Jira an "organizer," you may call Jira just a "ticket," and the fields could be different. In this space, how do you deal with the problem of semantic meaning of data? And how do you define success in terms of an onramp, assuming the company's ontology and your interpretation of it?

Adarsh Hiremath: The thing we've found is that understanding semantic meaning in a very large company data set is actually more or less a solved problem. Of course there are different optimizations you can make to improve recall. But for the most part, taking that data, splicing it appropriately, putting it in a high-dimensional space using whatever embeddings model you want, and then retrieving over it will give you very, very good results - especially when paired with a really smart model. So that understanding piece is something we support, but it's not as interesting to us as the second problem: how do you actually create the graph that represents all the relationships between the different data sources of a company?

It's less interesting to us to know what "Linear" means in a company's context versus Jira or organizer. It's how does that actually fit into the whole process of shipping a feature? What happens after Jira? Where does the collaboration for human work happen - is it Slack or something else? What does good collaboration look like when you're reviewing a feature that's about to be released? What does that review process look like? And when you're releasing the feature, what does the feedback loop look like if there's a big error and you want to find that error and maybe push a fix automatically? Tying all of those different steps together and making sure the knowledge graph for that company's context accurately represents relationships - that's the harder part.

Dev Tagare: And in this, we come back to the question of how time is not continuous. I may pick up a task today, do it over the next five days, kind of abandon it, pick it up again in two weeks. In your process of discovery you might find this as a high-impact task. Help us understand how you think about that process of discovery and tying it back to disaggregated steps -

and the shift from an agent solving for a goal to an agent sticking to a trajectory to get there.

Adarsh Hiremath: Yeah. Breaking apart that specific thing - sometimes humans will do a task in one sitting, they might wait five days. There are a couple of reasons for that. One is that the actual task is not that important and you need to reprioritize. Or you just don't have the bandwidth. The beautiful part of agents is that they don't have these constraints. They can prioritize and operate at exceptionally high throughput. When there are a bunch of different tasks that can be taken, the agent should be able to do them all simultaneously alongside high-value tasks that would be important in an enterprise context.

The piece that I think is unique and unsolved is the continuous learning and updating. How do you make sure the agent doesn't have a long feedback loop before it understands it made a mistake, or even understands that it could be doing better? That continuous learning loop involves a bunch of different things. It involves triaging tickets to humans who know what good looks like, if that's appropriate. In some cases it involves automatically understanding its failure from an LLM-as-judge. In some cases you could actually look at the telemetry of the API call and have that automatically improve the agent. So that continuous learning is a problem space I find very, very interesting and unsolved.

Nancy Wang: You know, when humans are doing workflows, you pause because you might not actually have all the context - even if you're human. Or you realize you need to tag in someone because they own a particular work stream. For the agent itself - because we often see hallucination - internally we have a no-code agent builder that allows our GTM teams to pull in contacts from across Drive, Slack, Notion, and so forth for their presentations. But for fields where the agent doesn't know something very confidently, it just makes up data. So how do you prevent that - where a natural human might pause, but the agent might just continue and finish the task?

Adarsh Hiremath: The key thing is citing the relevant company context. If the agent is stating a fact and is unable to cite the company context, it should automatically stop. That's functionality we already have in the Mercor platform - that's maybe level one. Level two is if it's able to cite the context but gets the wrong interpretation, you need to either have a human review the trajectory to create the eval, or have an LLM review the trajectory. Because some LLMs might be unable to get the right answer, but they might actually be better at reviewing the trajectory. You could arrive at the wrong answer but know that you arrived at the wrong answer and why. And you could arrive at the right answer and know that the reasoning was wrong and why. So you typically want multiple levels of review on the actual trajectory of the agent.

---

6. Precision vs. Conversational Latency

Dev Tagare: Yeah. Super cool. And in your observation, a lot of these incorrect answers or hallucinations tend to happen in what you'd describe as conversational agents - you're asking a question, getting an answer back quickly, and in the pursuit of quick answers, grounding is ignored, citations are ignored, models are sometimes underthinking or overthinking. Are you building towards a world where quality or precision takes over conversational experiences? There's a curve there - you create latency for accuracy, but eventually you want to bring accuracy to latency.

Adarsh Hiremath: Yeah. Addressing that - that's one of the reasons I'm most interested in evals as the foundational infrastructure investment that makes all of this possible. It allows you to select the best model for a specific use case. There is no quantitative, data-driven way to make that selection unless you have an eval suite that is representative of the job to be done and is comprehensive across the task, the trajectory, and the output. That's why we're so excited about evals and what they mean for the future. The models are getting to the point where the real bottleneck is applying evals throughout all the economically valuable parts of the economy, and then steering the models in the right way.

To your point about conversational interfaces versus using an agent as the backbone of a broader system - in conversational interfaces where the person is working directly with the agent, mistakes are actually a little bit more pronounced and easy to see. It's like the agent said something that doesn't quite make sense. But I actually think the more dangerous failures are the ones happening behind the scenes of these broader systems where you don't have a conversational interface to diagnose the failure. And often these failures happen silently and have much, much larger consequences than one would anticipate. The way to correct for that is an eval investment that is representative of the task that agent is doing.

---

7. Last-Mile Coverage, Trust, and Agent Authentication

Nancy Wang: And that actually brings us to a fundamental question that we think about a lot as a security company - trust and the ability to handle edge cases. In security, the difference between a false positive and a false negative is potentially your company ending up in headlines. So how do you think about last-mile coverage? Do you go for a broader set, or do you want to go for confidence on a very narrow slice of the problem?

Adarsh Hiremath: I typically say broader. The good part is that the models have reached a level of intelligence where, if they have all the right scaffolding, they can generalize. But you won't know unless you have a broad sample of the problem and you put that in a well-designed eval set.

On the topic of 1Password and trust and safety - one of the problem spaces I'm also really excited about is just authentication for agents. Humans need this to make sure their activities are secure. Agents need this as well. I think there's a very strong case to be made that the addressable market of agents taking actions that need to be gated appropriately is going to be larger than humans in the next couple of years, if it isn't already. So I'm very, very excited about that and what it means for 1Password and other companies focusing on security.

Dev Tagare: That's definitely blooming.

Nancy Wang: I will say I appreciate the call-out. Especially with agent swarms - you're already applying a multiplier effect on the human-to-agent ratio, and especially agents doing real work and taking real decisions.

Dev Tagare: Even just identity delegation - if you're talking to a third-party SaaS, you need a mechanism and you need coordination across. So yeah, that space is definitely booming. Related question, but slightly different from the security space - the context of an enterprise evolves over time. Today I may be a food delivery app, tomorrow I could do groceries, day after I could do something else, in the future drone delivery. So the context for an enterprise evolves. How do you keep that context evolution in perspective as you develop? And a related question: how do you solve for data drift?

---

8. Context Evolution, Data Drift, and Evals as a Living Investment

Adarsh Hiremath: Maybe just a broader point is that people often view investments in evals or whatever your infrastructure is as a one-time investment. I think that is extremely far from the truth. These are live investments. They're continuous investments. A good analogy is: if you're shipping a feature, you'd want to write unit tests for that feature. In the same way, if you're doing something with an agent, you want an eval set that covers the scope of that agent and evolves over time. A concrete example in customer support - let's say you have an agent trained to respond to inquiries about the product. But then a new feature is released. There needs to be steering and updating of the skills file to make sure that at the first time the agent fails on a query like that, there's automatically feedback inserted into the agent to address that new use case. And that's not just true for customer support - it's true for agents in all contexts. Companies are idiosyncratic. They evolve quickly. They add a bunch of new contexts. So that mechanism you use to distill the company context into the agent at high velocity and keep the feedback loop short is a very, very important one to solve.

Dev Tagare: Along the same axis - if you think about side effects that an agent would have in interactions with the model, or the model instructing something malicious back to the agent either directly or through other feedback loops - how do you think about adding a layer of security in the middle where you can enforce some policies and evolve the intelligence of the agent alongside the model?

9. Agent Safety and the Production Rollout Process

Adarsh Hiremath: Yeah. The first thing I want to call out is that it's really, really hard to roll out agents to production - in part because of this issue. That's supported by the fact that the initial adoption of these agents has largely been for personal assistant use cases where the human is always steering the agent. If the agent fails, the impact is on the order of magnitude of failing for one person, as opposed to maybe a million people. So that's a really, really hard problem that a lot of people are trying to solve - how to actually steer agents for production contexts that have a larger blast radius.

There's a natural progression of what you want to do as an agent development lifecycle. The first step is just trying the agent in a personal assistant context - does it feel like the agent can do the task? Then the next thing you want to do is connect it to some tools in a sandbox environment without real data. Validate that it can do the task again. Then you want to roll out a set of evals - live or not live - to more methodically see if it can do the task. Then you want to connect it all together in a sandbox environment that has real users, which means live evals connected to real tools, but not real data. The agent is actually usable. And then you want to do a phased rollout in a production context that has all of this infrastructure as well. The key is not jumping the gun and following the whole process.

Dev Tagare: Perfect. And then you put in a human-in-the-loop somewhere along the way, with and without production - that way you have some control over what could happen, what should happen, and so on.

Adarsh Hiremath: That's right.

---

10. Sandboxing and Remote Dev Environments

Nancy Wang: And as a quick aside - without wanting to take us down a huge rabbit hole - I was on a call with a large multinational bank this morning where they're saying that for more sensitive workloads, they're running it not just in sandboxes but in remote dev environments. Are you seeing that request from some of your more compliance-minded customers?

Adarsh Hiremath: Yes.

Nancy Wang: Yes. And what kind of workloads?

Adarsh Hiremath: Yeah. The typical ones are workflows in highly regulated industries - financial services-related use cases being a big one. The answer is yes. People like the general idea of not running the agent in production for the first time. You want to run it on sandbox environments, local and remote, just to see the whole thing under the sun. People are experimenting with that.

Dev Tagare: You know, sandboxing is going to be a big business very soon.

Nancy Wang: Yeah. I actually see the worlds of sandboxing and remote dev environments - like Coder, for example - converging over time.

Adarsh Hiremath: Totally. And we of course do a lot of that work with the labs - these RL environments. If you don't have access to the real production tools and context, or you don't want to give agents that access when you're rolling something out, it can be helpful to create a mockup. Select the agent, select the set of tasks you want to test, create mockups of all the tools the agent would need access to, use those mockups to roll out a trajectory, eval the trajectory, and eval the output. That's what people are doing in contexts like these.

---

11. The Apex Suite Benchmark: Measuring Real Work

Nancy Wang: So speaking of outputs - I want to take us to the recent benchmark you all published, Apex Suite. Just for listeners, Apex tries to measure whether frontier models can do economically valuable work over time. So what led you all to create this benchmark, and what was missing from existing benchmarks?

Adarsh Hiremath: The thing that inspired the whole Apex series of benchmarks is that we felt there are benchmarks that are really good at evaluating how well a model can do academic tasks - can it do IMO math, can it do tasks that are kind of an approximation of the economy, like a GDP-eval. But there's no benchmark that can really evaluate how well a model can do enterprise work output, or knowledge work, or just real work.

That's when we started with Apex - the first version, which was largely for professional services. We made that benchmark authentic. And then we also rolled out Apex Suite in the software engineering context. The biggest gap I see is that a lot of these benchmarks are isolated to small patch tests - okay, this whole repository is messed up, just patch these functions. It focuses on patches as opposed to the whole system. And secondly, the failures you see in coding models based on these benchmarks - the failure of the benchmarks to create a distribution of real software engineering work - is also the failure you see in the models in production.

So Apex Suite is our attempt at addressing that. There are two types of tasks in Apex Suite. The first is observability - in a real enterprise context, if a system fails, you would need the agent to triage across all these different things, go look at the logs, say "hey, this is not right," and then raise that to the user. And the second is integration - not just a patch test or a single repository, but a set of systems and how they integrate with one another. And the last thing about Apex is that correctness - whether the code compiles or not - is not the only thing. It's also: is this code you'd be excited about being in your codebase six months from now? Does it follow style requirements? So the way we actually come up with the benchmark score for Apex Suite is a rubric that is not just correctness-based, but also has other criteria to evaluate how well models are actually performing in a real context.

Dev Tagare: One of the themes that comes out of the paper is that strong performance seems to come from what you'd call optimistic discipline - knowing that you've verified something versus just producing it, and continuing to verify it over time. So how does this discipline look in the world of agents, where now you have swarms solving for competing goals like humans would in a big company? And how does that benchmark evolve to address an ecosystem problem versus a specificity problem at the level of an individual engineer?

Adarsh Hiremath: I would say this is very top of mind for us, and we're actively working on a project that is super relevant to this. The short answer is: the unexplored space is agents as coworkers and agents actually talking to each other. We have benchmarks that evaluate how well an agent can do a task when given a task. But do we have benchmarks that evaluate how well agents can work with one another? The answer is no. And it's very top of mind for us.

Dev Tagare: And similarly - taking the same concept a little further - external environmental factors change. That creates a level of uncertainty. Humans react to it in a certain way. Agents are more deterministic - or there are instructions associated with it.

Adarsh Hiremath: There are no feelings.

Dev Tagare: There are no feelings. You could say that.

Nancy Wang: More predictable.

Dev Tagare: More predictable, for sure. So when uncertainty hits or environmental factors change, how are you thinking about evolving this ecosystem to account for it? And how would you measure how the agent did versus how the human did?

Adarsh Hiremath: I know I sound like a broken record, but the answer is just the eval evolution. In the same way that you can scaffold the task, trajectory, and output, you can do the same thing with a human. You can get as granular as literally recording what they're doing - the audio, the video, the screenshots, all the tools. Or you can use documents, or do moderated interviews. But there are ways to create a similar infrastructure for humans and then directly compare that with agents via evals.

Dev Tagare: Makes sense. And the feedback loop there is a little bit latent - delayed, if you will. You think something like a policy evaluator, where you have a stated skill for what your company's policies or approach is - let's say you codify those into a skill, there's your thinking, there's your principles, there's your operational style - and that keeps running in the background grading what's happening. Recently there's news that Meta has done something similar with Zuck's personality or skills. Do you think that could be an interesting way where the company takes on its own skill, the enterprise takes on its own skill, and that becomes like a continuous feedback loop - agnostic of events?

Nancy Wang: I have a vested interest in that question, because actually my CISO and I have both created - let's call them Nancy and Jacob skills - which we've shared with the rest of the company in terms of "what would our CISO think about this problem?" or "what would Nancy say about this problem?"

Dev Tagare: I have my own swarm of agents that I run personally. Yeah.

---

12. Democratizing Knowledge Through Agent Personas

Adarsh Hiremath: I mean, the answer is that I think this is very much a possible thing. And as Nancy mentioned, 1Password is already doing this. More broadly, the thing that's interesting to me is that agents present a very unique opportunity to democratize and diffuse knowledge across a company. Without these agents, without a CEO agent, people operate in the silo of their specific organization, team, or KPI that they're hill-climbing. That can cause problems - misaligned incentives, not necessarily out of malice, but just out of a lack of context. But in a world where you have the Nancy agent, and the engineering team might have a question about Nancy's overall strategy and 1Password's overall strategy, you could just ask the agent. It has access to all the documents that Nancy has and becomes a very easy way to democratize and diffuse that context in an organization. I feel very strongly that what we're seeing with the Zuck agent, the Nancy agent, and a bunch of others will become the norm in these companies.

Dev Tagare: One related technique I'm trying out is like, "hey, what's top of mind for you?" So on a weekly basis, I've been marking meetings or summaries of meetings as top-of-mind categories. So when someone triggers "hey, what's top of mind for Dev?" they actually get that response updated on a weekly basis. No one really needs to ping me for anything. They can just go to your chat space and add me, and that shows up. It's a cheap way to communicate information.

Nancy Wang: Yeah. Like, really scales your time.

Dev Tagare: Yeah. It definitely does.

Adarsh Hiremath: In some cases, agents can be better at structuring my thoughts than I can be, because they're able to actually process a bunch of information that I'm not able to process. Or even if the conclusions are the same, there might be something novel that we can also learn from. So I find these agents to be great assistants and companions.

Dev Tagare: And coworkers.

Adarsh Hiremath: Companions, coworkers.

Dev Tagare: They could be companions at some point.

Nancy Wang: Well, not one of the first wave of AI companion companies. Yeah. Yeah.

---

13. From Token Economy to Outcomes Economy

Dev Tagare: One of the things I was thinking about - you mentioned earlier that you want to bring agents to every surface, where work happens. Because where work happens is super contextualized - very in-the-moment, might last one day, two days, five days, and so on. You reconcile that with your Apex benchmarks, and now you're talking about the actual productivity gain for a given individual, for a given enterprise. So do you think we're moving away from a token-based model to a value-added model to an outcomes-oriented model - like the outcomes economy versus the token economy?

Adarsh Hiremath: Absolutely. I think the reason people are so fixated on token consumption today is because outcomes are pretty hard to measure. Also, agents aren't able to operate completely autonomously and produce work output without a level of steering - so how do you attribute the value to the agent versus the person who steered it? That can be a little bit tricky.

But I think the value-at-stake model will become more and more the norm when AI agents fully take on the role of employees at companies - where you'd judge how well a person does based on their work output, and similarly you'll judge how well an agent does by its agent output. It just feels inevitable that we're going to go down this way. And the most exciting applications to me are the ones that on net increase productivity, as opposed to just cost savings. I think of the AI wave as a massive lever on the productivity of society. It's super exciting to me what people will be able to achieve - what abundance will be created as a result of it - as opposed to just incremental cost savings, as important as those are.

Nancy Wang: And bringing this back to your benchmark - what struck us when we were reviewing Apex was the fact that most benchmarks really focus on things that are easy to measure. When we talk about doing actual work that a human would do, there are all these edge cases, all these nuances. And the fact that we may not make the same decision on the same amount of data tomorrow or even yesterday - it's really in that moment. So how did you think about balancing simplicity for your benchmark versus reality?

Adarsh Hiremath: I would say it's definitely skewed towards the reality side. If you look at these benchmarks, you can tell they're simple because they're saturated. And you can tell they're not realistic because we aren't seeing adoption across all of these different companies. To me, it's a multi-trillion dollar problem. These agents are smarter than me at a bunch of different things. Smarter than a bunch of the smartest people I know. But they're not doing a lot of knowledge work tasks that I know they're capable of doing - in part because they don't have the enterprise context, enterprise tool connections, and feedback loop for what good looks like. The real way this all plays out is these are brilliant employees being dropped into a company with no onboarding. Solving that problem is a multi-trillion dollar problem, because it will enable agents to operate at high throughput and high sophistication across a bunch of different use cases and increase the productivity of humanity.

---

14. Mercor as the System of Record for Agent Behavior

Nancy Wang: So maybe bring it home a little bit. What result would you be looking for from this, so we can get more mass adoption across builders?

Adarsh Hiremath: The benchmark as it stands today - the frontier labs are actively looking at the data samples and hill-climbing the benchmark. At a base model level, I'm very, very excited that this benchmark will make the models more capable of doing real engineering tasks in the context of Apex - specifically the two categories we're excited by: observability, where there's a production failure, you have to go through all these different systems to figure it out and then actually resolve it; and integrations, not just a single system but a set of different systems. I'm really, really excited that that intelligence is going to be present in the base models, and hopefully all of us will feel it when we're using the next generation of coding models.

The second thing is making it accessible and useful to a broader set of people. There's a little bit of work in taking the base data set and making it contextual to an enterprise's context. Every enterprise is idiosyncratic - if that was not the case, those companies would not exist. Every single company is doing something special, or at least believes they are. And a benchmark that reflects the nuances of that enterprise, the way they work, and the tools they use is going to be really important in making coding agents the best at that company's SDLC.

Dev Tagare: That's a great point. So switching gears a little bit - Mercor sits at a very unique spot between the human network and now what we'd call the agent swarms or the AI network, evolving into more of an operating system for agents overall. Breaking this down, how much of the moat do you think is infrastructure versus product and services versus user experience and the data flywheel? Usually it's a combination of things, but relatively speaking - six months out, a year out, two years out - how do you think that categorization is landing?

Adarsh Hiremath: The way I think about it is: the system of record for specifying agent behavior. You want to be able to answer two questions as well as possible. One: what should the agent do, given the company context? Two: is it doing it well or not - which is the evals that identify whether there's a failure, and then the continuous learning loop to make sure that failure never happens again.

If we solve those problems really, really well, then there's a reason for companies to keep using us. There's also this last point - I really believe that there are certain models that are really good for specific use cases. As the spend on inference increases by 1000x, picking the right model for the right use case is going to become more and more important. And evals are the answer to that. A representative eval set allows you to make those optimizations, even if you're just focused on cutting down your cloud bill. You need a quantitative way to understand which models to use. So to me, it really, really comes down to evals. I believe every company will create an eval set that is representative of all the different jobs that their people are doing, all the different departments, all the tasks to be automated. And there will be a whole set of good things that come out of that.

Nancy Wang: And to your point - as people use these purpose-built models more and more, they're going to generate more eval sets which are going to further fine-tune these models. Enterprises are thinking about: how much context do I actually need to give in order for this to be super purpose-built for my environment? So how do you counter questions from enterprise customers who are like, "hey, maybe I don't want FTEs to annotate those data sets," or "I don't want it to index that data store"?

Adarsh Hiremath: Well, it depends on the specific appetite of that company. There are some companies that will happily use a model like Gemini and have faith in the security of that model and the retention policies. They say, "we want to stuff everything in there" because they recognize that if you enable a really smart model with the right context to do more, spreading that across the company might triple the market cap, because you're able to do so much. There are other companies that, rightfully so, have very specific privacy, trust, and security requirements. In which case they might opt to use an open-source model - either from one of the established labs or newer labs - and have everything local. They'll still put in that context, but it won't be leaving the company's premises at all.

Dev Tagare: Do you think in this space, enterprise context actually becomes the moat? Because if the model gets more intelligent over time, what you really control is the harness and the context that goes with the harness. If Mercor is effectively building the harness and curating the context and the agent trajectories with the evals, that then becomes a repeatable offering, agnostic of the models that are getting better.

Nancy Wang: From there, essentially your platform offering.

Adarsh Hiremath: Exactly right.

Nancy Wang: And how do you share that safely across - let's say there's some context you glean from 1Password that might be applicable across different security or infrastructure companies - how would you securely share that?

Adarsh Hiremath: Well, the answer is we don't share it. They're not shared. There are silos at both an infrastructure level and an operational level. Infrastructure meaning every tenant is a different one, separated and treated as such. And the second thing is from an operational standpoint - trust with our customers is everything. You would not want FTEs working with a specific company to also be working with a competitor. So we take both of those components pretty seriously.

---

15. Builder's Mindset and Hot Takes

Dev Tagare: Want to do the Builder's Mindset?

Nancy Wang: Yeah.

Dev Tagare: Let's do it.

Nancy Wang: Yeah. So with the Builder's Mindset, we like to end the episode on some spicy hot takes. My spicy hot take is - and we've talked about this at length - actually trustworthy agents can exist in the next 12 to 18 months.

Adarsh Hiremath: Agents can exist in 12 to 18 months, definitely. It sounds like your take is that they're not going to?

Nancy Wang: Or - actually, I'd say let's hear yours.

Dev Tagare: My take is I think agents will become more like orchestrators - leaner and leaner over time. The action will shift partly towards more of the harness and more to the models as the models become more intelligent. What we think of as agents today will be like a dispatcher or an orchestrator, effectively.

Adarsh Hiremath: Okay. But then the answer would be yes - agents will still exist. It's just they'll be doing more orchestration.

Dev Tagare: Purely just doing orchestration. Everything else will look more like traditional software engineering, feeding back to the models.

Adarsh Hiremath: But an agent does have to do the software engineering.

Dev Tagare: A little bit, yeah. I mean, today we began with goal-driven agents and a long iteration flow, where the number of tokens you'd need to get to the outcomes you want was super high and the feedback loop was kind of latent. My thesis is that because you will begin with evals first and the models can do deep retrievals simultaneously, you'll go from a largely goal-driven system to a trajectory-driven system governed by a thin orchestration layer.

Nancy Wang: So what would that actually look like? Just a set of instruction sets?

Dev Tagare: A set of skills and tools, effectively. And then like a general arbiter of picking skills and tools and bootstrapping what we call agents today. But there's no random goal-driven thing anymore - it's all evals-driven, evals from trajectories. Models are able to keep the task on a trajectory.

Adarsh Hiremath: I would say I largely agree with that. Even if you just take the agent-to-agent interaction out of the picture - one of the things that people said about agents that was kind of incorrect is that it's a unidirectional thing between humans and agents. You give the agent the task, it just does it. It's very much bidirectional. The human gives the agent the task. It needs human feedback to understand what good looks like, to not make mistakes, all the things we've talked about. And I think the natural progression of that is the bidirectional

collaboration model with agents - where an agent will need to talk to another agent, orchestrate with another agent, oversee the output of another agent's work. I think that's certainly something that will happen in 12 to 18 months.

Dev Tagare: It's effectively becoming a three-way handshake between the human, the set of agents, and maybe the models as a separate category. And the models are more a constellation of models versus a single model.

Nancy Wang: Or a mixture of experts.

Dev Tagare: Mixture of experts. Yeah. So that's where I think 2026 is the year of the model again. 2025 was probably the year of the agents and the swarms.

Nancy Wang: That's something to revisit in December.

Dev Tagare: Quite likely going to be wrong given how things are evolving. But we'll see.

Nancy Wang: And we also love to end the episode with: if you had a month to build anything you wanted - it could be Mercor, it could also be something completely different - what would that be?

Adarsh Hiremath: I feel like I have to say Mercor, in part because it's true, and because I have an obligation for it to be Mercor. But one of the things I wish I could spend some time on is just random side projects. You can't help but walk around and have ideas pop into your head with the pace at which things are evolving. So maybe a little bit of a Sunday here and there, I'll make some time to build out some projects.

Nancy Wang: Awesome. I'd love to showcase one on our episode.

Dev Tagare: Let's do it. Yeah. Thank you. Appreciate the time. This is a lovely conversation.

Adarsh Hiremath: Thank you for having me.
