

Towards AI · [Follow publication](#)

★ Member-only story

CTO's ADAPT System: My Five Bets for the Agentic Engineering Era

A Principal's Bets for Thriving in the Agentic Engineering Era

7 min read · Oct 5, 2025



Kapil Viren Ahuja

Following ▾



Listen



Share



More

We lost a project. A big one. During the pitch, my team's estimates seemed high-risk-buffered as always. But with AI agents already transforming engineering velocity, those numbers felt like artifacts from a different era. We didn't get the project. That loss woke me up. More importantly, it needed to wake up my team.

In 2025, I transitioned from overseeing Digital Experience Transformation for our clients as a CTO to a hands-on Principal, building our next-generation agentic systems. As CTO, I made dozens of strategic decisions weekly. As Principal, I made one: build systems that make better decisions than I could.



I immersed myself in the chaotic reality of the Agile AI for SDLC. I am building the ‘Phoenix’ (agentic SDLC system), which is enhancing our development lifecycle, potentially reducing our microservice development time from 35 days to 3.5 days. Additionally, I have begun building the Storytelling Engine, which will provide us with a composable Agentic System to utilize within the Digital Experiences framework. I’ve seen firsthand what works, what’s just hype, and what will define the next era of software engineering. These bets aren’t disconnected; they form a cohesive foundation for my strategy over the next 12 months. Each bet supports the others, creating a flywheel of acceleration.

• • •

Bet 1: Build Agents That Build Agents

While many developers are now using AI assistants — and that's a great start — the real competitive advantage comes from creating your own. The distinction is critical: transitioning from being a user of AI to a builder of AI systems is what separates top performers from the rest.

My goal isn't to be the best engineer with agents — it's to make my entire team better than me. Based on my experience building agentic systems like Phoenix, I estimate that engineers who can design, build, and deploy custom agents are already operating in the top 1% of the industry. They are not just writing code faster; they are building scalable, repeatable problem-solving machines.

Why the big jump?

Custom agents let you tackle problems at a speed that was previously unimaginable. When you codify a situation into an agentic system, you create leverage that compounds infinitely. Gartner predicts that 40% of enterprise applications will feature task-specific AI agents by 2026 — up from less than 5% in 2025. This isn't incremental change; it's an 8x explosion in one year. The organizations building custom agents now are positioning themselves at the leading edge of this transformation.

My first custom agent took me three weeks to build. It automated a task that previously took 20 minutes to complete manually. For six months, I'd spent more time maintaining it than I'd saved. I almost gave up. Then I realized: I wasn't measuring the right thing. The value wasn't the 20 minutes — it was the repeatable system I could now scale infinitely.

This is the fundamental shift: agents aren't about doing more of the same work faster — they're about creating exponential leverage through systems that learn, adapt, and compound value while you sleep.

. . .

Bet 2: Unlearn Your Way to Relevance

As engineers, we're constantly learning new things, but this is the first time we need to be willing to erase what no longer serves us.

The significant shift in thinking is this: it's no longer about what you can do, but what you can *teach your agents* to do. The engineers who thrive won't be those who cling to mastery of outdated skills — they'll be those who aggressively prune their expertise to make room for agent orchestration, systems thinking, and strategic judgment.

The risk here is obvious: breaking habits is hard. It's uncomfortable. We must also accept that some of our skills will atrophy. Think about how calculators affected our ability to do mental math, or how GPS changed how we navigate. However, I bet that we will not need the skills that will atrophy. The payoff in terms of productivity and future-proofing our careers will make it well worth it.

. . .

Bet 3: Systematize the Agentic Workflow

If you've worked with agents, you've hit the two main bottlenecks: planning and reviewing. For a significant part of the next six months, humans will continue to be the limitations that prevent us from addressing these inefficiencies. However, I'm betting that by late 2026, these limitations will be smashed.

The 2025 DORA Report, for instance, found that while AI boosts throughput, it also correlates with “increased instability” in AI-heavy teams, mainly because the review process for AI-generated code is significantly more challenging.

We can generate new plans and hand them off to our agents faster than ever, but the planning process itself still relies on us. And when the agent is done, we're the ones who have to review the work. These are the bookends of the agentic coding loop, and they're where things slow down.

I'm betting big on custom agents working together in a pipeline — an agentic development workflow. This combination of old-school engineering knowledge and new-world agent power is the most powerful abstraction we currently possess.

. . .

Bet 4: Build High-Bandwidth Interfaces

Let's be honest, the chat interface is the most overused and least imaginative way to interact with agents. In 2026, I will be focusing my time and energy on multi-agent UIs and new interaction interfaces, particularly voice-based ones.

The goal is to increase the bandwidth between you and your agents. How can we accomplish more work in a shorter amount of time? By commanding more compute with better interfaces. Imagine being able to articulate your ideas and have a team of agents develop them for you.

Voice-driven tools like WisprFlow represent the emerging class of high-bandwidth interfaces. In testing such systems, we've found that verbal articulation can increase agent orchestration speed by 50%, suggesting significant potential for workflow transformation.

The challenge is significant. How do you design a UI that orchestrates multiple agents without confusing the user? My team says there will soon be invisible UIs. It's a complex UI/UX problem, and an excellent opportunity for designers who enjoy building. We're currently prototyping agent orchestration through communication platforms like Discord, exploring whether existing collaboration infrastructure can serve as an effective interface for multi-agent systems.

. . .

Bet 5: Maximize Compute

My final bet is that we must become "compute maximizers" — relentlessly focused on using as much compute as possible to solve valuable problems. The reality is, however much computing you think you're using, it's not enough.

Unless you have an agent working for you 24/7, you can and should be using more. And then you should not stop until you have two of them. My goal for 2026 is to have an always-on agent, and I encourage you to consider this a strategic imperative.

This doesn't mean jumping from simple prompts to a massive, 24/7 system overnight. It's a gradual process of identifying opportunities to offload cognitive work to the machine.

Unless I use \$100 a day, I am not happy. The question I ask every day is, "How can I use more compute?" Think about all of my developer workflows, my documentation

workflows, and anything and everything that is still untouched. The potential is immense.

The risks are real: cost, infrastructure complexity, and the potential for wasted cycles. You must invest resources to understand what's possible. The ultimate reward is living software that works for you, even when you're not — a truly autonomous extension of your engineering capability.

My compute investment runs approximately \$4,000 annually; roughly 1% of our innovation budget. This generates estimated productivity gains of 20%, suggesting strong ROI for compute-intensive strategies. The question for leaders isn't whether this is expensive, but whether the competitive disadvantage of underinvestment is acceptable.

. . .

Conclusion

Building Your ADAPT System

Yes! System, not Systems (Singular)

I call this the ADAPT System because it captures what's really required: building custom Agents as a core competency, aggressively deprecating the skills that no longer serve us, systematizing our Agentic workflows into repeatable Pipelines, creating high-bandwidth interfaces that Power our orchestration, and relentlessly maximizing compute to Throttle up our problem-solving capacity. The five bets I've outlined are not a Chinese menu; they are an interconnected, interdependent strategy for survival and growth. Building custom agents (Bet 1) is pointless if you don't deprecate the old skills to make room for them (Bet 2). A systematic workflow (Bet 3) enables scaling beyond the single principal mindset; however, that scale is useless without high-bandwidth interfaces (Bet 4). And a commitment to utilizing more computing power to solve more valuable problems (Bet 5).

I've transitioned from being Michael Jordan — the best player on the court — to being Phil Jackson: the coach who builds championship systems. This requires a fundamental shift in how I think about my role, my skills, and the very nature of software development and everything I do. It's a journey I'm on myself, and it's one filled with both frustration and immense reward.

So, where do you start?

Start Small: Don't try to boil the ocean. Pick one repetitive, high-value task in your current workflow and design a simple agent to automate it.

Embrace the Learning Curve: Your first agents will be clumsy. You will spend more time debugging the process than you save. The Next one will be very different. This is expected. The goal is not immediate productivity; it's long-term capability.

Think in Systems: As you build, constantly ask yourself: "How does this scale? How does this connect to the rest of our workflow?" The goal is not just to build an agent, but to build a system of agents.

The lost project that started this journey taught me something crucial: the gap between where we are and where we need to be isn't about technology — it's about courage. The courage to admit that our hard-won expertise might be our most significant liability. The courage to invest in systems that make us feel obsolete. The courage to build agents that might be better than us. That's the transformation ahead. The only question is: will you lead it, or will it leave you behind?

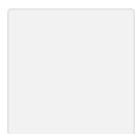
AI

Agentic Ai

Thought Leadership

Leadership

Future Of Work



Follow

Published in Towards AI

90K followers · Last published just now

Making AI accessible to 100K+ learners. Find the most practical, hands-on and comprehensive AI Engineering and AI for Work certifications at academy.towardsai.net - we have pathways for any experience level. Monthly cohorts still open—use COHORT10 for 10% off!



Following ▾

Written by Kapil Viren Ahuja

95 followers · 38 following


No responses yet



Bgerby

What are your thoughts?

More from Kapil Viren Ahuja and Towards AI


 In Generative AI by Kapil Viren Ahuja

Top-Down AI Adoption Beats Bottom-Up Every Time (Except When It Doesn't)

I have been watching leaders struggle with AI adoption for years now, and I need to tell you something that might sound contradictory...

Sep 15 🖱 57




 In Towards AI by Teja Kusireddy

We Spent \$47,000 Running AI Agents in Production. Here's What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16 🖱 2.5K 💬 80




 In Towards AI by Ashish Abraham

No Libraries, No Shortcuts: LLM from Scratch with PyTorch

The no BS guide to build, train, and fine-tune a Transformer architecture from scratch

★ Oct 2 🖱️ 1.1K 💬 14



 In CodeToDeploy by Kapil Viren Ahuja

CTO Playbook for AI-Native SDLC: The Landscape

We all need one. I am building mine in real-time


★ Aug 18 🖱️ 5 💬 1



See all from Kapil Viren Ahuja

See all from Towards AI

Recommended from Medium


 Jettro Coenradie

Spec-driven development using Codex and Backlog.md

Don't worry, this is not one of those blogs telling you we no longer need developers. In my daily work as a developer, I have become...

Oct 11  4



 Marcin Ros

The Rise of AI-Native Apps: How Software Itself Is Being Rewritten

For decades, apps were built around menus, buttons, and clicks

Oct 1




 Dave Patten

Spec-Driven Development: Designing Before You Code (Again)

AI has changed how we build software - developers now code alongside assistants that analyze repos, write tests, and design architectures...

Oct 13  52  2



 In CodeToDeploy by Kapil Viren Ahuja

CTO Playbook for AI-Native SDLC: The Landscape

We all need one. I am building mine in real-time



Aug 18



5



1



Aakash Gupta

Amazon Just Fired 30,000 People After \$60B in Profits. Here's What They're Not Telling You.

Amazon is laying off 30,000 workers.

2d ago



48



2





🌸 Ines Zenkri 🌸

My Coffee, My Code, and the Day Amazon's AI Dream Crashed

It was a regular Tuesday morning. I was at my desk, sipping my coffee and deep in the flow state, building an agent with Amazon Bedrock...

1d ago



See more recommendations