January 9, 2026 | Chris Lema

Most Multi-Agent AI Collaboration is Faking It.

I remember the first time I did a job interview with a panel. It was intense. Each person was reacting from their own perspective, looking from their own background, either nodding in agreement or tilting their head with a pending question. No one was waiting their turn. No one asked permission to observe. They were...

AI · Agents · For Engineers

I remember the first time I did a job interview with a panel. It was intense. Each person was reacting from their own perspective, looking from their own background, either nodding in agreement or tilting their head with a pending question. No one was waiting their turn. No one asked permission to observe. They were all watching me simultaneously, and the lead interviewer was synthesizing their reactions in real time.

That's the picture in my head when I think about multi-agent collaboration.

And it's nothing like what most "multi-agent frameworks" actually do.

The Lie Everyone Is Selling

Look at this code and tell me what you see:

```
`const research = await researchAgent.run(query); const analysis = await analysisAgent.run(research); const summary = await summaryAgent.run(analysis);`
```

That's not collaboration. That's three function calls in a trench coat pretending to be agents.

Or the slightly fancier version:

```
`const result = await orchestrator.run([ { agent: 'researcher', input: query }, { agent: 'analyst', dependsOn: 'researcher' }, { agent: 'writer', dependsOn: 'analyst' } ]);`
```

That's still not collaboration. That's orchestrated function calls with agent cosplay. Each "agent" waits its turn. The "orchestrator" is just a for-loop with better marketing.

Here's the test: if you can replace your multi-agent system with a simple function chain and lose nothing, you don't have agents. You have a pipeline wearing a costume.

```
`function doWork(query) { const research = research(query); const analysis = analyze(research); return summarize(analysis); }`
```

Same result. Less complexity. No reason to call anything an "agent."

Why This Matters More Than You Think

Here's the thing about real collaboration: time doesn't wait in line.

Think about that panel interview again. While I'm answering a question about my technical experience, one interviewer is evaluating my depth of knowledge. Another is watching the clock because we only have 30 minutes. A third is thinking about team fit. They're not taking turns. They're all observing simultaneously, and they might have conflicting signals.

The technical evaluator wants me to go deeper. The timekeeper knows we need to move on. The culture person just noticed something in my body language.

In a sequential system, here's what happens:

`Candidate responds → Wait for TechnicalEvaluator to finish thinking (2 seconds) → Wait for Timekeeper to calculate → Wait for CultureEvaluator to assess → Now the lead interviewer decides`

By the time the Timekeeper "gets its turn," 2+ seconds have passed. Its observation is stale before it's even considered. The moment is gone.

This is what happens in most agent frameworks. They process sequentially and pretend they're collaborating. But real scenarios

don't wait for your agents to take turns:

- Time pressure doesn't pause while your depth evaluator makes an LLM call
- Conflicting signals need to be weighed simultaneously, not sequentially
- Fast observations shouldn't wait for slow ones
- Synthesis happens on incomplete information, not after everyone reports

If your agents take turns, they're not collaborating. They're standing in line.

What Real Collaboration Actually Looks Like

I've been building a demo to prove this difference. It's a tutoring scenario: a student gives a book report about "A Wrinkle in Time," and four agents evaluate the conversation. Five minutes. Five topics. Real-time synthesis.

Here's what happens when the student responds in a parallel system:

```
`Student responds ├── DepthExpert starts async LLM call ├── Coordinator opens 800ms collection window │ │ [Meanwhile, every 10 seconds, independently:] │ Timekeeper: "We have 3:42 left, pressure is medium" │ │ [DepthExpert finishes:] │ DepthExpert: "Rating 2/3, recommend probe" │ │ [Grader reacts to DepthExpert:] │ Grader: "Running grade B+, 2 topics scored" │
```

└── [800ms window closes] Coordinator synthesizes ALL observations that arrived`

The Timekeeper isn't waiting for permission. It's observing reality on its own schedule. The DepthExpert doesn't know or care about the Timekeeper. They both observe and publish. The Coordinator listens to everyone.

And here's where it gets interesting. What happens when the DepthExpert says "probe deeper" but the Timekeeper says "we're running out of time"?

In a sequential pipeline, this conflict can't even exist. By the time one agent finishes, the other hasn't run yet. There's no simultaneity, so there's no tension to resolve.

In genuine parallel collaboration, the Coordinator receives both observations in the same window and makes a judgment call: "We have enough time to probe, but keep it brief" or "Skip the probe, move to the next topic."

That's collaboration. That's synthesis. That's what happens in a real panel interview when two evaluators exchange a glance and the lead interviewer makes a call.

The Architecture That Actually Works

The pattern is simpler than the sequential pipeline once you see it:

`Events (reality) → Multiple independent observers → Coordinator synthesizes → Action`

No agent-to-agent messaging. No waiting in queue. No orchestrator deciding who goes next. Each agent subscribes to events that matter to it, observes independently, and publishes what it sees. The Coordinator has a collection window, takes what's available, and decides.

The 800ms window is the key insight. The Coordinator doesn't block waiting for the DepthExpert to finish its LLM call. It takes what's available and decides. If DepthExpert is slow, the decision happens without it. If it's fast, its input gets included.

That's collaboration under uncertainty. That's how real teams work. You don't wait for everyone to finish their analysis before making a call. You synthesize what you have in the time you have.

The One Question That Exposes Fake Agents

Here's a diagnostic you can use on any "multi-agent" system, including your own:

Can multiple agents observe the same event simultaneously and independently decide to act?

If yes, you have genuine collaboration.

If no, you have a pipeline with agent names.

Most frameworks fail this test. They're built on request/response architectures. One thing happens, then another thing happens, then another. There's no "simultaneously." There's no independent observation. There's just a sequence with better branding.

The frameworks that pass this test are built on different foundations. Actor models. Pub/sub architectures. Systems designed for concurrency from the ground up, not bolted on as an afterthought.

What This Means For What You're Building

If you're evaluating agent frameworks or building multi-agent systems, you have a choice.

You can use the sequential approach. It's easier to reason about, easier to debug, and works fine when time isn't a factor and your agents don't need to observe the same events simultaneously. Just don't call it collaboration, and don't expect it to scale when reality gets complicated.

Or you can build for genuine parallelism. Multiple observers. Independent awareness. Synthesis under time pressure. It requires different architectural foundations, but it actually does what the marketing promises.

The panel interview in my head, the one where everyone is observing and reacting and the lead interviewer is synthesizing in real time, that's not a pipeline. That's not turn-taking.

That's what real multi-agent collaboration looks like.

And most frameworks aren't even close.

Where I'm Building This

I've been working with Jido and Jido AI on Elixir, and it's the first stack I've found that delivers genuine parallel collaboration out of the box. Elixir inherits Erlang's actor model, which was built for telephone switches where millions of concurrent processes needed to observe and react independently. Turns out that's exactly what multi-agent AI systems need.

Each agent is its own process. They subscribe to events via PubSub. They observe independently. They publish what they see. The Coordinator synthesizes. No turn-taking. No orchestrator deciding who goes next.

If you're hitting the wall with sequential "agent" frameworks and want to see what real collaboration looks like, check out Jido and Jido AI. The architecture is different. The results are different. And once you see it, you can't unsee how fake the alternatives are.

I won't send you spam. And I won't sell your name. Unsubscribe at any time.

ABOUT THE AUTHOR

Chris Lema has spent twenty-five years in tech leadership, product development, and coaching. He builds AI-powered tools that help experts package what they know, build authority, and create programs people pay for. He writes about AI, leadership, and motivation.

AI IS MOVING FAST. YOU DON'T HAVE TO FIGURE IT OUT ALONE.

I help business leaders cut through the hype and put AI to work where it actually matters.

Let's Talk AI

CHRIS LEMA

Making Experts Dangerous

Tools

- Your Voice Profile
- Your Audience Segments
- Your Content Agent
- Your Platform Profile
- Your Lead Magnet Agent

Content

- Blog
- Book
- Speaking
- Coaching

Connect

- Contact
- About
- LinkedIn
- X / Twitter
- YouTube

Privacy • Terms