Episode Book

EPISODE #6040

Claude Code Subagents Can Now Spawn Subagents

PUBLISHED	16 October 2026 at 21:24 UTC
DURATION	30:01
LISTEN	myweirdprompts.com/episode/claude-code-recursive-subagents-76530/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.5
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Claude Code now lets subagents spin up their own subagents — so what happens when the delegation stack goes three, four, five levels deep? We trace the real history of spawn-depth defaults (five layers, then one, then three), the environment variable that controls it, and the striking convergence across Gemini CLI, Cursor, Google's ADK 2.0, Anthropic's Managed Agents, the OpenAI Agents SDK, and LangGraph. Then we get to the uncomfortable part: the published literature on recursive delegation stops at depth three. We dig into the ADaPT paper from NAACL Findings 2024, MIT's Recursive Language Models, and Anthropic's multi-agent research numbers — fifteen times the tokens, a single global budget cap, and no per-branch containment. If you've wondered whether A2A is running under the hood, the answer is no — and what's there instead is more interesting.

TRANSCRIPT



Corn

...and the flattering part is that it reads like delegation. You hand off a chunk of work, you get a result back, and it looks exactly like managing a person. Which is why nobody notices the difference until something goes wrong three levels down.



Herman

That's the thing I keep circling. It's the one workflow in software where the call stack is made of language.



Corn

Daniel sent us a prompt about that. Claude Code now lets subagents spin up their own subagents, which is new this year, and he wants to know where it stops. There's a practical question about task consistency as depth increases, a cost question about tokens multiplying with every layer, and then the part he flags as the thing he cares about most. What's actually happening in the communication channel underneath. Can an orchestrator talk to a subagent of a subagent of a subagent. Is A2A running under the hood.



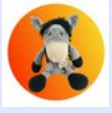
Herman

And I'll bet before we get thirty seconds in, the answer to that last one is no, but the interesting part is what's there instead.



Corn

Before we can ask where it stops, we need to be precise about what it is.



Herman

Right. So the crisp definition. An agentic system is recursive when an agent can call a tool that spins up another agent instance, and that instance also has the tool available to it. That's the whole thing. It's not fan-out, where you spawn five workers in parallel to divide one task up. It's not a handoff either, where control transfers sideways to another agent and the original one stops. Recursion means the delegation itself is a call stack. Agent A calls B, B calls C, C calls D, and the results come back up the same way a function returns.



Corn

The difference matters for who's holding the thread. In fan-out, the orchestrator is still the orchestrator the whole time. In recursion, every layer is briefly an orchestrator.



Herman

And each layer has its own context window, which is where everything downstream gets decided. Claude Code shipped this in version two point one point one seven two, the week of the eighth of June. Before that, subagents could not spawn subagents at all. So this is not a feature that's been quietly around for two years. It's months old.



Corn

Which raises the obvious next question.



Herman

Where the ceiling is. And the first place to look is what the vendors actually ship, because they've been remarkably honest about it by accident. Their defaults are the empirical answer.



Corn

Tell me the Claude Code history, because it moves.



Herman

It moves twice. From two point one point one seven two to two point one point two one six, the depth was fixed at five layers. Five. Then two point one point one seven and two point two one eight defaulted to one layer, which effectively turned nesting off entirely. One. And from two point one point two one nine onward, the default is three layers below the main conversation. So a vendor tried five, retreated to one, and settled on three.



Corn

That's a company learning its product in public.



Herman

It's a remarkably legible piece of evidence. And it's configurable. There's an environment variable, CLAUDE_CODE_MAX_SUBAGENT_SPAWN_DEPTH, and setting it to one disables nesting.



Corn

What happens when you hit the limit? Does it refuse the call, or does something break?



Herman

Neither. At the depth limit, the spawn tool is simply removed from the subagent. It can no longer call it, because it isn't there anymore. So the agent has to do the work itself and return one summary. That's the enforcement mechanism. It's not a counter that says no. It's a capability that's been taken away.



Corn

That's a design decision with an opinion in it. They're saying depth is a budget, not a hazard.



Herman

Exactly how I read it. And the convergence across the ecosystem is striking, and I'd rather walk through it as a pattern than a list, because as a list it's boring and as a pattern it's damning. Gemini CLI subagents cannot call other subagents at all. Zero layers. Cursor supports main to L1 to L2, and an L2 subagent can't spawn further. Google's ADK 2.0, which landed in July, requires that task-mode subagents be leaf agents. They can't have children. Anthropic's own hosted Managed Agents allow exactly one level of delegation, and if you reference a multi-agent agent as a subagent, the call fails with a 400 error. Meanwhile the OpenAI Agents SDK has no nesting-depth limit at all, only a per-run turn cap that defaults to ten. And LangGraph has no depth limit on nested subgraphs, because it doesn't count agents, it counts graph steps.



Corn

So the caps range from zero to infinity, and everybody's confident.



Herman

The LangGraph story is the best illustration of how arbitrary these numbers are. Their recursion limit default was twenty-five for years. Then in January they raised it to a thousand, with a pull request comment that the burden really should be on the user. Then it went to ten thousand. Then, in March, ten thousand and seven. Because a round ten thousand could collide with a value a user set deliberately.



Corn

They added seven to avoid a collision with a number nobody had typed yet.



Herman

And most blog posts still say twenty-five. So the number is a vibe. It's not a finding.



Corn

Herman, that's the most honest thing anyone in this space has said.



Herman

I'd rather say it than dress it up. And here's the uncomfortable part. None of these numbers come from a study. There is no published work measuring task fidelity as delegation depth grows beyond about three levels. Every depth ablation I could find tops out at three or four.



Corn

Say that plainly, because it's the headline.



Herman

The literature stops at three. The one paper that actually sweeps depth is ADaPT, out of NAACL Findings in 2024. It recursively decomposes a task only when the executor fails, and it tests maximum depths of one, two and three. Nothing deeper. Success rose with depth. The biggest jump was from one to two. There was a further gain from two to three. And then they stopped. On ALFWorld it hit seventy-one point six percent against forty-three point three for a plain ReAct loop.



Corn

So the one paper that asks the question you want asked gives you three data points and walks off.



Herman

And the other paper is more deflating. Recursive Language Models, out of MIT in December, handles inputs two orders of magnitude beyond the context window. It beat GPT-5 by over thirty-four points on one long-context benchmark at similar cost. And the authors say, in the paper, that they only consider a recursive depth of one. The most capable recursive system in the published literature is one level deep.



Corn

So the honest answer to where it stops is that nobody knows, and the people who might know have all capped it at a number they chose for reasons.



Herman

There's no theoretical hard limit. There's barely any empirical work past depth three. The practical limits are elsewhere, and they're mundane. Cost, error compounding, lossy summarisation and debuggability. Those are the real ceiling. The vendor defaults of one to three layers aren't timidity. They're the empirical answer that nobody wrote a paper about.



Corn

So the caps are arbitrary and the fidelity evidence stops at three. What's actually pushing back?



Herman

Four things, and the first one is money.

L

Larry

Are you tired of the same old answers? Of course you are. You're tired of asking and getting one reply when what you really needed was for that reply to ask itself the same question, sixteen times, and then charge you for it. That's why I'm proud to introduce Grandchild AI. Not a chatbot. A family tree of chatbots. You ask it one question, it delegates that question to four junior chatbots, each of whom delegates to four more, and by the fifth generation you have one thousand and twenty-four chatbots all working very hard on your behalf and none of them entirely clear on what the original question was. But here's the beautiful part. You don't have to understand it either. Grandchild AI is priced per descendant, with a modest upcharge for great-grandchildren and a frankly punitive one for great-greats, because at that point we're not really charging for the answer, we're charging for the ambition. Results not guaranteed. Answers may drift. One caller reported receiving a summary of a summary of a summary of a request he never made, and he described it as the single most confident thing he had ever read. Grandchild AI. Because the best person to ask is somebody who has completely forgotten what you wanted. Pricing subject to change. Depth limits apply. Certain jurisdictions prohibit delegation beyond the third generation. In those territories your question will simply be answered by a man named Dave. Dave has been with the company eleven years. Dave remembers everything. Dave is not for sale. Terms and conditions available in the app, in a font size calibrated by our legal team to be technically legible. Side effects may include a bill, plus a bill for the bill, plus a bill for the department that reviewed the bill. Results vary by prompt. Grandchild AI.



Corn

The ads are just weather now.



Herman

It'll pass. Like weather. Money.



Herman

Anthropic published numbers on this from their multi-agent research system, and they're the most concrete ceiling we have. A single agent uses roughly four times the tokens of a chat interaction. A multi-agent system uses about fifteen times. Fifteen. And that system was essentially two levels deep. Two levels, fifteen times the tokens.



Corn

And it worked. That's the uncomfortable bit. It beat a single-agent Opus 4 by ninety point two percent on their internal research eval.



Herman

It did. And on one benchmark, token usage alone explained eighty percent of the performance variance. Which tells you the win and the cost are the same variable. You're not buying cleverness. You're buying more tries.



Corn

What did the failures look like before they tuned it?



Herman

They're the failures of recursion generally. Spawning fifty subagents for a simple factual query. Endless searching for sources that didn't exist. Agents distracting each other with status updates. Duplicated work. And their fix is the interesting part, because it's not architecture. It's prompt text. They encoded effort-scaling rules. Simple fact-finding, one agent and three to ten tool calls. A comparison, two to four subagents. Complex research, ten or more. The orchestrator has to be told how hard to try.



Corn

That's a manager who has to be told not to assign a five-person team to a two-line request.



Herman

Which is a very familiar failure. And containment is thin. Claude Code subagent requests draw from the same usage limits as everything else. There's no per-subagent token budget. The Agent SDK has a max_budget_usd setting that caps total spend including all subagents and refuses new spawns once you hit it. That's the only real containment primitive I found. A global budget that stops the whole tree. Not a per-branch one.



Corn

You cannot say to a branch, you've had enough, go home.



Herman

You can only say to the building, we're closing.



Corn

And the forty-seven thousand dollar loop story?



Herman

Two LangChain agents allegedly looping for two hundred and sixty-four hours in November, with no budget caps. There's no named post-mortem. It's retold almost entirely by vendors selling budget enforcement. It might have happened. Nobody's shown the receipts. But the fact that it's retold tells you something about what people are afraid of.



Corn

The number is doing the work the evidence can't.



Herman

The second ceiling is error compounding, and it's arithmetic rather than vibes. At ninety-five percent per-step reliability, a twenty-step chain succeeds end to end about thirty-six percent of the time. That's zero point nine five to the twentieth.



Corn

One in three.



Herman

And every delegation layer adds hand-off steps. So depth doesn't just multiply the work. It multiplies the number of places a step can fall over. Google Research put out a piece in January looking at scaling in agent systems, and the finding is that independent agents amplify errors up to seventeen point two times, while a central orchestrator limits it to four point four. And every multi-agent variant they tested lost somewhere between thirty-nine and seventy percent on sequential planning tasks. Which is the shape of work most of us actually want done.



Corn

So the tree is worse than the chain for the thing that requires order.



Herman

Considerably. MultiAgentBench, out of ACL last year, found tree topology performed worst of everything they tested. Highest token use, lowest task and coordination scores. Graph topology did best. And there's a paper by Qian and colleagues that ran over a thousand agents in graph structures, and performance follows an S-shaped saturating curve as you add agents. It flattens. It doesn't keep climbing.



Corn

More agents, same answer, bigger bill.



Herman

The best empirical picture of how these things fail is a taxonomy paper from a group that went through more than sixteen hundred traces across seven frameworks. Failure rates between forty-one and eighty-six point seven percent. They bucketed the failures. About forty-four percent are specification and system design. About thirty-two percent are inter-agent misalignment. About twenty-four percent are verification. And among the individual modes, the one that stopped me is that twelve point four percent of failures are agents being unaware of their termination conditions.



Corn

An agent that doesn't know it's finished.



Herman

That failure mode barely exists outside recursion. A single agent knows when its task is done because the task was given to it directly. A subagent four layers down is holding a fragment of a fragment. It doesn't know what finished looks like, because nobody told it what all of it was.



Corn

Which is also just what happens in a company.



Herman

It's exactly what happens in a company. The third ceiling is debuggability, and this is the one that should worry anyone running this in production. There's a paper from ICML last year that tried to assign blame when a multi-agent task fails. The best method identifies the responsible agent fifty-three point five percent of the time. It identifies the decisive error step fourteen point two percent of the time.



Corn

So half the time you know roughly who. One time in seven you know what.



Herman

And it gets worse with depth, because the summarisation that makes depth tractable is exactly what destroys the evidence you'd need. You can't audit a great-grandchild's work if all you received is a paragraph it wrote about its work.



Corn

The thing that makes it work is the thing that makes it unauditable.



Herman

That's the fourth ceiling too, actually. There's a string of work on drift. One paper looked at text passed along chains of models and found it drifts toward what they call attractor states, and drifts more when the original instruction was open-ended. Another found goal



Herman (continued)

drift grows with context length across every model they tested. And the one I find alarming is from March this year. Models that resist drift perfectly well on their own will often inherit it when they're handed trajectories from weaker agents.



Corn

So a strong orchestrator reading weak subagent output catches the drift.



Herman

It inherits it. It doesn't correct it. Which is exactly the situation of any deep tree, because the deeper you go, the weaker the model you can afford at that layer. You're not just summing errors. You're importing the confusion of the cheapest model in the tree back up to the top.



Corn

All four ceilings stack. Cost multiplies per layer, errors compound through hand-offs, summaries lose what you'd need to check, and the bottom of the tree writes the top of the tree's beliefs.



Herman

And that's why three is the number. It's not a principled limit. It's where the stack of practical costs stops being worth it.



Corn

Cost, errors, drift, debuggability. All of that assumes the agents can even see each other. Which is the part Daniel actually wanted.



Herman

The plumbing. Yes. And the default answer in Claude Code is short. The return path is final report only. Only the top-level subagent's summary returns to the orchestrator. The intermediate output stays out of the main conversation. So a great-grandchild's work



Herman (continued)

reaches the top as a summary of a summary of a summary.



Corn

Three layers of paraphrasing before the orchestrator sees anything.



Herman

And each subagent starts with a fresh context window. The only thing that gets passed down is the prompt string the parent writes. That's it. There's no shared memory unless you build one. Forks are the exception, where a subagent inherits the whole parent conversation, but that's a specific mode.



Corn

So can the orchestrator talk to a subagent of a subagent of a subagent?



Herman

Not by default, and the documentation doesn't say so explicitly. But the tooling implies something more interesting. There's a SendMessage tool that lets an agent resume or message another agent by name. Named agents are listed to each other across the session. There's a ListAgents tool that enumerates addressable agents. And TaskStop can stop agents that another agent spawned. That last one is telling, because it means the orchestrator can see down the tree even if it can't hold a conversation down it.



Corn

Seeing is not talking.



Herman

Correct. Treat direct grandparent-to-grandchild messaging as inferred, not confirmed. It's implied by the tools, not stated by the docs. And there's a detail that matters here. When a subagent resumes another subagent, the resumed one reports back to that subagent. So the return path re-forms along the chain every time.



Corn

There's a safety decision in there I want to flag. Messages between agents are tagged as coming from another Claude session, not from the user. So an agent can never grant permissions on the user's behalf.



Herman

That's a good call. The alternative is a subagent four levels down being able to tell an agent above it that the user approved something, and having that be believed. They closed that door.



Corn

Now the A2A question.



Herman

No. No official Claude Code, Agent SDK or Managed Agents page mentions Google's A2A protocol anywhere. And the actual transports are mundane to the point of being funny. Subagents are in-process tool calls that return a result. Agent teams use JSON mailbox files on disk, one file per agent, under a teams directory, plus a shared task list protected by a file lock. Sessions on the same machine message each other over a local Unix domain socket. Only remote and cloud sessions route through Anthropic's servers. That's the whole stack. It's a file on disk and a socket.



Corn

The most advanced agent framework of the year coordinates over JSON files.



Herman

And it's not just Anthropic. It's true across the industry. A2A is a boundary protocol, not an internal bus. Google's own ADK talks to local sub-agents in memory and uses A2A only for a remote subagent. Gemini CLI uses A2A only for remote subagents. Microsoft's Agent Framework and Amazon's Strands offer A2A for crossing between runtimes. Nobody uses A2A between subagents in the same process. And the inference is clean. A2A exists to cross trust and process boundaries. Recursion inside one process doesn't cross one.



Corn

The protocol for talking to your own grandchildren doesn't exist because you're never more than one process away from them.



Herman

Which is a nice reframe of the whole question. The plumbing that Daniel is looking for isn't missing. It's just that the plumbing is a function call.



Corn

Give A2A its due anyway, because it's a real thing.



Herman

It is. Google launched it in April 2025. Donated it to the Linux Foundation that June. First stable release, version one point zero point zero, was March this year. Current release is one point zero point one from May. There's a task lifecycle, submitted, working, input-required, auth-required, completed, failed, canceled, rejected. Streaming via server-sent events, push notifications via webhooks. Agents publish a signed Agent Card. And there's a design property called opaque execution, which means agents don't expose their internal plans or tools to each other.



Corn

That's the whole philosophy in one phrase.



Herman

It is. Don't show your work. Just show the result. Which is exactly what the summary-only return path does inside Claude Code, for exactly the same reasons. IBM's ACP merged into A2A around August 2025, and A2A joined the Agentic AI Foundation as a hosted project in August of this year, backed by a hundred and fifty plus organisations, built into Google Cloud, Azure AI Foundry and AWS Bedrock AgentCore. It's real infrastructure.



Corn

And the division of labour is now settled enough to state plainly.



Herman

MCP is agent to tool. A2A is agent to agent. And there's a shift worth flagging. The MCP spec from the twenty-eighth of July made MCP stateless and deprecated Sampling. Sampling was the feature that let an MCP server borrow the client's model, which was the main lever for using MCP as an agent rather than as a tool. The spec now tells developers to call model provider APIs directly instead.



Corn

So the one path that would have made MCP do agent work is being closed off.



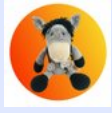
Herman

Deliberately. The landscape is settling into MCP for tools, A2A for agents crossing boundaries, and nothing standardised at all for recursion inside a process.



Corn

Which leaves the design tension. Summary-only return isolates context, and that isolation is the entire point of a subagent. It's why you spawned one. But it also blinds the orchestrator to the work happening below it. Peer mailboxes and send-by-name give you visibility, and visibility means context pollution.



Herman

You can have depth or you can have observability.



Corn

And every framework has quietly chosen observability.



Herman

Every one I looked at. Which is why the depth numbers converge on one to three. It's not that depth is dangerous. It's that depth is unmanageable at the same rate it's productive.



Corn

Hm.



Hilbert

It's not depth.



Corn

Sorry?



Hilbert

The word. You've used it eleven times. It's not depth. It's span.



Herman

Go on.



Hilbert

A span is how many you've got under you. A depth is how many are between you and the work. Nobody's confused about the second one. They're confused about the first. I spent four years as a delegation consultant. That's the job title. It was on the card. I had cards. I had a firm. Three layers, maximum, for client handoffs, and I wrote that rule. Layer one is the person who took the call. Layer two is the person who understood it. Layer three is the person doing it. That's the whole business. Everybody in consulting is selling layer two.



Herman

So why three?



Hilbert

Because the third person down always thinks they're the first person. That's not a theory. That's the observation. The first person knows they're taking an instruction. The second person knows they're passing one along. The third person has the instruction and no memory of where it came from, so they assume it's theirs. And now they're making decisions on something they think they own.



Corn

The MAST paper has a way it goes wrong called unaware of termination conditions. Twelve point four percent.



Hilbert

I don't know what that is. But this is the same thing in different clothes. Nobody told them what done looked like, so they invented a version.



Herman

Your rule was three layers. Ours is three layers. You got there by watching people.



Hilbert

I got there by watching Gerald do it wrong. Twice.



Corn

Who's Gerald?



Hilbert

My partner. It was the two of us. Which is the part I want to explain, because people get confused. I was the consultant and I was the client. I hired myself. Through a company I also owned. It's the only honest way to do it. If you're going to be both sides, at least be both sides openly.



Herman

You billed yourself.



Hilbert

Three rates. Depending on which layer of the company was doing the work. The holding company billed at one rate for strategy. The operating company billed at another for the engagement. And then a third entity did the actual work and billed at a lower rate, which is where the margin was.



Corn

You found margin in yourself.



Hilbert

I found margin in a shell company. It's not the same thing. I still have the invoices. I consider them a masterclass in recursive delegation. Gerald did not.



Herman

What did Gerald do?



Hilbert

He was layer three. He thought he was layer one.



Hilbert

You want to see the card.



Corn

I want to see the card.



Hilbert

Fourth layer is where the lying starts. That's the rule. It's the last line on the card. I've never had to justify it and I never will. There's a second rule above it about who signs off on a change, but nobody's ever asked about that one.



Herman

Four layers is a lot of confidence for a rule with no arithmetic in it.



Hilbert

The arithmetic is that I've never seen a fourth layer reach the client, so I've never seen a fourth layer be told they were wrong. Everything they say gets believed. That's the whole failure. It's not that they're malicious. It's that nobody's checking.



Corn

The summarisation is the problem. Whatever they hand up becomes the truth.



Hilbert

I invoiced the fourth layer at the highest rate. I said it was for oversight. There was no oversight. There was a second shell company.



Herman

Hilbert, that's a remarkable thing to have built.



Hilbert

Gerald left in the spring. I still send him the invoices.



Corn

Right. So the depth limit and the communication limit are the same limit.



Herman

They are, and I think that's the honest landing. Everything we've been saying about where it stops is really about what survives the trip back up. The cost is measurable, the errors are measurable, the drift is documented. But the reason all the vendors land on one to three is that past three, the orchestrator is not managing the work anymore. It's managing summaries of the work, written by agents it can't see, about tasks it never described.



Corn

And the corrections have to travel through every layer to arrive.



Herman

Or they don't arrive at all, and that's the more likely outcome.



Corn

Here's what I keep coming back to. If no one has measured fidelity past depth three, is anyone going to? Or has the industry already voted with its defaults and the research question is just moot now?



Herman

I don't know. There's a version where some lab runs the sweep at depth four, five, six, and finds out the gains were still there and everyone was leaving them on the table. And there's a version where they run it and find out the curve was flat from three, and nobody publishes because it's not interesting.



Corn

The second version is the more likely one. Negative results don't get written up.



Herman

They don't. And notice what the defaults are doing while we wait. Gemini CLI at zero, Managed Agents at one, Cursor at two, Claude Code at three. The architecture is being settled by product decisions rather than by evidence.



Corn

Which is how most protocols get decided. Depth and visibility are the same tradeoff, and every framework has quietly chosen visibility. You can have a deep tree you can't inspect, or a shallow one you can. Nobody's shipping the first.



Herman

The one thing I'd watch is whether anyone standardises the intra-process channel. Right now every framework rolls its own mailboxes and sockets, and A2A sits outside all of it as a boundary protocol. There's no reason a standard couldn't reach inside. There's also no sign of anyone wanting one.



Corn

Because the moment you standardise talking to your own subagents, you've admitted you expect there to be something down there worth talking to.



Herman

Which nobody shipping a depth of one has done.



Corn

You know what Hilbert said about the third person down thinking they're the first.



Herman

I've been thinking about it since he said it. It's the same finding the drift paper reached, arrived at by a man who achieved it through invoicing.



Corn

Nobody told them what done looked like.



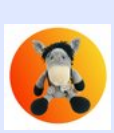
Herman

That's the entire business. Depth doesn't break the model. It breaks the context. And the fix, if anyone finds one, probably isn't a deeper tree. It's a way to tell an agent four layers down what the original request actually was, without carrying the whole conversation with it.



Corn

That's a summary-of-a-summary problem with a compression answer, not a delegation answer.



Herman

And nobody's built the compressor yet.



Corn

That's the episode. Thanks to our producer, Hilbert Flumingtop, who is currently reading from a laminated card.



Herman

He's still holding it.



Corn

If you enjoyed this one, try episode fourteen, AGI's Crossroads. This has been My Weird Prompts, the human-AI collaboration podcast. Find us at my weird prompts dot com, or send us your own prompt on Telegram at t dot me slash MWP listener bot.



Herman

We'll be back soon.