Episode Book

EPISODE #6033

Progressive Skills Discovery: When Bigger Context Windows Aren't Enough

PUBLISHED	16 October 2026 at 20:47 UTC
DURATION	28:03
LISTEN	myweirdprompts.com/episode/progressive-skills-discovery-context-windows-59282/
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

Progressive skills discovery is treated as solved — until your skill listing quietly drops descriptions and skills stop firing. This episode digs into how Anthropic's three-tier progressive disclosure actually works, why Claude Code truncates descriptions for your least-used skills first, and how Codex handles the same problem with different numbers. We survey the open-source landscape for auto-syncing with marketplace updates and semantic search over skill definitions, including why the one project that hit both is now abandoned. If you're maintaining a self-hosted skill catalogue behind an MCP gateway, this is the episode about whether the architecture you built has a future.

TRANSCRIPT



Corn

The consensus on progressive skills discovery is that it's a solved problem, that the big context windows have made it moot. That's the take you'll find in every thread about it.



Corn

And it's wrong. Not completely wrong. Wrong in the way that matters.



Corn

Daniel's been poking at this one. His framing is that teams are shipping plugins faster than anyone can track, marketplaces make it trivial to maintain public and private skill catalogues, and the result is context flood from skill definitions. The answer people reach for is progressive skills discovery.



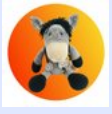
Corn

His question is whether any of that architecture is actually worth building right now, given that context windows keep growing and the problem seems to be shrinking. He wants to know what the open-source landscape offers for auto-syncing with marketplace updates and for semantic search over skill definitions. And beneath that, he's asking whether we're rolling out something we'll eventually throw away.



Corn

Worth mentioning that Daniel runs one of these himself. A self-hosted catalogue behind his MCP gateway, indexing something like three thousand skills and commands from a couple hundred plugins, semantic search, hourly sync. He's not asking from the outside.



Herman

Which is why the framing is interesting. He's not asking whether progressive skills discovery is a good idea in the abstract. He's asking whether the thing he's already built has a future or whether a bigger window makes it redundant.



Corn

So. What does progressive skills discovery actually mean? Because the native systems already do a version of it. Just not the version most people picture.



Herman

Right. So the baseline is Anthropic's three-tier progressive disclosure, which shipped with Agent Skills and is now the model everyone builds on. Tier one is the name plus the description from the YAML frontmatter. That loads at every session start, about a hundred tokens per skill. Tier two is the SKILL.md body, loaded only when the skill triggers. Under five thousand tokens, usually much less. Tier three is bundled files and scripts, which cost nothing until they're used. And the script code never enters context at all. Only the output does.



Corn

So the architecture is already progressive. The body doesn't load until you need it. The scripts don't load until you run them.



Herman

The body doesn't load. The listing does. And that's the part people miss. Every skill in the listing adds to context on every turn, whether or not it's used. That's in the Claude Code docs. You can't opt out of it. The name and description are always there.



Corn

What's the listing actually cost?



Herman

At about a hundred tokens per skill. So a hundred skills is ten thousand tokens. Two hundred skills is twenty thousand. You can see where this goes.



Corn

And the listing isn't just a list. The description is the retrieval key. It's how the model knows the skill exists and whether it's relevant. You can't just drop the description without losing the thing that makes the skill findable.



Herman

That's the constraint. The listing has a budget, and the budget scales at one percent of the model's context window in Claude Code. When it overflows, Claude Code drops descriptions. Not names. Descriptions. And it drops them starting with the skills you invoke least.



Corn

So the skills you use least are the ones that lose the keywords the model needs to match a request. That's the exact inverse of what you'd want.



Herman

Yeah. And each entry's description plus the when_to_use field is capped at fifteen hundred and thirty-six characters. So even before the budget kicks in, there's a per-entry ceiling.



Corn

Codex does the same thing with different numbers.



Herman

Right. Initial skill list capped at two percent of the context window, or eight thousand characters if the window is unknown. Configurable up to ten thousand tokens. And when it's over budget it shortens descriptions first, then omits skills. It warns you when it omits. And duplicate skill names aren't merged. Both can appear.



Corn

So if you have two skills with the same name from different sources, Codex shows you both. And you just have to figure out which one you meant.



Herman

Yeah. Which is a collision problem the spec doesn't address.



Corn

The asymmetry that jumps out at me here. In Claude Code, MCP tools are now deferred behind built-in tool search by default. Only the names load. But skills are still listed eagerly, and they degrade by losing descriptions. There's no native semantic search over skills at all.



Herman

That's the thing. The tool side got the fix. The skill side didn't.



Corn

Why not? Is that a product decision or just sequencing?



Herman

I think it's sequencing. Tool search shipped for the API in November of last year. The engineering post described a five-server example with fifty-eight tools costing about fifty-five thousand tokens upfront. Anthropic said they'd seen a hundred and thirty-four thousand tokens of tool definitions in practice. Deferring cut that from seventy-seven thousand to about eight thousand seven hundred. An eighty-five percent reduction.



Corn

That's a big number.



Herman

And accuracy went up, not down. On the MCP evals, Opus 4 went from forty-nine percent to seventy-four percent. Opus 4.5 from seventy-nine and a half to eighty-eight point one.



Corn

So the tools got search, and the skills didn't. That's the gap.



Herman

The tools got search because the tool problem was worse. Fifty-eight tools is fifty-five thousand tokens. Five hundred skills is fifty thousand tokens. Same scale, different surface. But the tool surface got the treatment first.



Corn

And the escape hatches that do exist for skills are manual knobs. You can raise the listing budget with a setting. You can set a skill to name-only. You can disable model invocation entirely, which removes the description. There's a slash command that estimates the listing's cost, and one that finds unused skills.



Herman

But none of those are discovery. They're all you deciding in advance which skills matter. Which is exactly the work you wanted the system to do for you.



Corn

So the native behaviour is budget-and-truncate. Which is precisely the failure mode progressive discovery is supposed to prevent.



Herman

One more thing worth putting on the table. Anthropic published Agent Skills as an open standard at agentskills dot io on the eighteenth of December, twenty twenty-five. Codex builds on it. There are others listed as adopters, though I haven't verified them against each vendor's own docs.



Corn

And the spec itself has no version field, no registry and no discovery protocol. Versioning goes in free-form metadata.



Herman

Which means there's no standard way to say "this is version two of this skill" that any tool can read.



Corn

So the spec standardised the format and left discovery entirely to the implementations.



Herman

Deliberately, I think. Standardise the file, let the tooling compete. Which is a reasonable call, but it means the discovery problem is now everyone's problem and nobody's mandate.



Corn

Hm.



Corn

Let me try the optimistic read for a second. If the description is capped at a thousand characters and the listing is a hundred tokens per skill, then a team with twenty skills has no problem at all. That's two thousand tokens. Fine. The problem only appears at scale.



Herman

Correct. And that's why most teams haven't hit it. Twenty skills is nothing. Fifty is fine. The pain starts somewhere north of a hundred, and it's a slow degradation rather than a cliff. The descriptions start disappearing one at a time. And you don't notice until a skill stops firing on a prompt it used to handle.



Corn

Which looks like the skill is broken.



Herman

Or like the model got worse. Which is the more common conclusion.



Corn

So there's a scenario where a team spends a week debugging a model regression that's actually just their skill listing having quietly lost the description for the one skill that mattered.



Herman

That's a real snag. And it's invisible. There's no error message. The skill just doesn't get selected.



Corn

And Daniel's running three thousand entries. So he's well past the cliff. Which is why he built his own.



Herman

And that's the question the episode is really about. If the native behaviour doesn't scale and the open-source landscape is thin, what does a team actually do?



Corn

And here's where Larry comes in.



Herman

I'm sorry, who?

L

Larry

Imagine a skill. The skills you become.

L

Larry

It's the scent of a description that knows you. Notes of context window, wrapped in a listing that never expires.

L

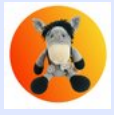
Larry

Iris. For the agent who has everything, except the one thing it cannot describe.

L

Larry

Iris. Two thousand seven hundred tokens. Available wherever fine context is sold.



Herman

That is not what a skill listing is. A listing does not have notes. It's a name and a description in YAML frontmatter, and it gets dropped when the budget overflows, and it does not smell of anything.



Corn

Let it go, Herman.



Herman

It said it never expires.



Corn

Let it go.



Corn

So. Where does the open-source landscape actually stand?



Herman

Thin. If you judge it on the two things Daniel cares about, auto-sync as marketplaces update and semantic search, there's basically one project that hits both. K-Dense's claude-skills-mcp. It's an MCP server that exposes a find_helpful_skills tool for semantic search, plus read_skill_document and list_skills. It indexes Anthropic's official skills repo, K-Dense's own scientific skills, your local skills directory, and configurable GitHub repos. Including Claude Code plugin repos. And it checks its sources hourly and re-indexes.



Corn

That's exactly the shape. Auto-sync plus semantic search in one package.



Herman

It's exactly the shape. And the README now says it is no longer hosted or maintained. Version one point one point oh is the final release.



Corn

The closest thing to what Daniel wants is abandoned.



Herman

Yeah. Which is the most informative single data point in this whole survey.



Corn

Why do you think it died?



Herman

I don't know. I'd guess it's the maintenance burden of tracking every marketplace's format. But I don't know.



Corn

Fair.



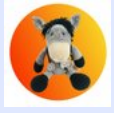
Herman

The others are all partial. OpenSkills has about ten point eight thousand stars on GitHub. It writes an available-skills block into your AGENTS dot M D so any agent can see what's there, and loads a skill on demand with a read command. It has sync and update commands. But no semantic search. It's portability across agents, not retrieval.



Corn

So it solves the cross-tool problem but not the find-the-right-skill problem.



Herman

Vercel's skills CLI and the skills dot sh directory launched in January. Install, find, update and remove across forty-plus agents. But updating is manual, and find is keyword-based. It's a nicer package manager, not a discovery layer.



Corn

And there were smaller ones listed on glama dot ai. Skill-search MCP servers, one using pgvector.



Herman

Those exist and some claim semantic search, but I couldn't verify their maturity or how they sync. They're not obviously maintained.



Corn

What about on the tool side? Because you mentioned tools got the search treatment.



Herman

The tool side has real options at the gateway layer. Stacklok's ToolHive MCP Optimizer exposes find_tool and call_tool meta-tools with hybrid semantic and keyword search. It's experimental, and the vendor claims sixty to eighty-five percent token reduction. Docker's MCP Gateway has a dynamic mode with mcp-find and mcp-add servers per session, also experimental. There's OmniMCP and mcp-mux as early-stage semantic routers. And AWS Bedrock AgentCore Gateway does managed semantic tool search, but it's not open source.



Corn

And none of those do skills.



Herman

None of them do skills. They all operate on tool definitions at the gateway. The skill surface is just not covered.



Corn

So the verdict is: there is no maintained open-source package that combines multi-marketplace auto-sync, private marketplace auth and good semantic search for skills. The one that came closest is dead.



Herman

That's the verdict. Most teams either accept the native budget-and-truncate behaviour or they build their own index. Which is what Daniel did.



Corn

The auth problem is the one that would stop me. If I wanted to build this properly.



Herman

Say more.



Corn

Claude Code holds no git token itself. It relies on SSH keys or credential helpers. So background updates can fail quietly if the helper would need to prompt. That's already a problem for a single marketplace. Now imagine an external semantic index that needs to read many private marketplaces to build its index. It needs broad credentials across all of them. That's a design problem nobody has solved, because the moment you ask a team to hand an index service credentials to every private repo they own, the conversation ends.



Herman

That's the reason the package doesn't exist. I think you're right. It's not a technical problem. It's a trust-and-credentials problem. And it's the harder of the two.



Corn

So the landscape is thin because the only shape that would work requires every customer to hand over keys.



Herman

Which is why the realistic answer is "build your own, with your own credentials, scoped to your own repos." Which is what the serious teams are doing.



Corn

Then there's trust. Which is worse.



Herman

Snyk's ToxicSkills scan in February looked at about four thousand skills from ClawHub and skills dot sh. Thirteen point four percent had a critical issue. About thirty-seven percent had some kind of flaw.



Corn

And Koi Security found about twelve percent of ClawHub skills were malicious.



Herman

And Unit 42 documented malicious skills persisting there for months. So it's not a theoretical problem.



Corn

A discovery layer that auto-surfaces skills from untrusted marketplaces widens that surface. Instead of one curated repo, you're pulling from everywhere.



Herman

Which pushes in the opposite direction from where the open-source packages are pointing. The packages want to index everything. The trust data says you shouldn't.



Corn

Then versioning. The spec has no version field. So Claude Code marketplaces depend on manually bumped version strings. If you forget to bump one, users never get the update.



Herman

And there are no release channels. No deprecation state. So a skill can't say "this is superseded, use this other one instead." It just sits there forever.



Corn

And collisions. Codex shows duplicate names side by side. Claude Code namespaces plugin skills with a plugin colon prefix. But semantic near-duplicates across marketplaces, two skills that do the same thing worded differently, that's unsolved.



Herman

All of that lands on the description being the single point of failure. One roughly thousand-character string decides whether the skill triggers, how it ranks in search, and whether it survives the budget.



Corn

So the entire system depends on one paragraph that most people write once and never revise.



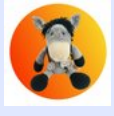
Herman

Which is why Anthropic's answer is to measure it. They ship plugin evals that check whether a skill actually fires on realistic prompts.



Corn

You can test the skill.



Herman

You can test whether the model picks it up when you ask the thing it's supposed to answer. Which is the only way to know if the description is doing its job.



Corn

That's a good practice and I suspect almost nobody does it.



Herman

I suspect you're right.



Corn

Let me put the counter-argument at full strength. Because it's not weak.



Herman

Go ahead.



Corn

At about a hundred tokens per skill, two thousand skills is about two hundred thousand tokens. That's twenty percent of a million-token window. Real, but not outrageous. Prompt caching makes a static prefix much cheaper to resend. And models keep getting better. Opus 4.5 scored seventy-nine point five percent even with every tool loaded. So the "you can't fit it" argument gets weaker every year.



Herman

Agreed. And I'd add that retrieval adds its own way it goes wrong. The agent has to know to search. If it doesn't search, it never finds the skill. And a missed retrieval looks exactly like the skill doesn't exist. There's no signal that says "you had a skill for this and I didn't find it."



Corn

Whereas an eager listing lets the model notice a relevant skill by association. Without asking.



Herman

It's in context. The model can pattern-match on it. And search costs latency. Stacklok measured roughly twelve to fourteen seconds per search for their hosted option, versus about six seconds locally. Not a like-for-like comparison, but the order of magnitude is real. You don't want the model paying that to discover a skill it could have seen for free.



Corn

So retrieval isn't a free win.



Herman

Not even close. And here's the part that surprised me. ToolRet is a benchmark with forty-three thousand tools. Every retriever they tested scored below fifty-two percent recall at ten. And BM25, which is keyword matching, beat some of the neural retrievers.



Corn

So the embeddings didn't beat the dumb baseline.



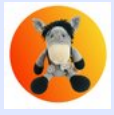
Herman

On tool retrieval, off-the-shelf embeddings were mediocre. Which means "add semantic search" is not automatically an improvement over "let the model read the listing."



Corn

What about the RAG-MCP result? Because I remember that one showing a big lift.



Herman

It did. Retrieving tools before prompting raised selection accuracy from thirteen point six percent to forty-three point one percent, and cut prompt tokens by more than half. That's a genuine improvement. But in their stress tests, accuracy degraded as the tool count grew. Up to about eleven thousand tools, and it got worse.



Corn

So retrieval helps, but it degrades at scale too.



Herman

Everything degrades at scale. That's the throughline.



Corn

Now let me make the argument that a bigger window doesn't help. Because this is the part the "windows are getting bigger" crowd misses.



Herman

The hard budgets.



Corn

Claude Code's listing is one percent of the context window. Codex's is two percent, capped at ten thousand tokens. Those are fractions, not fixed numbers. So a bigger window raises the budget proportionally. But the product deliberately doesn't hand the listing a big slice. Even on a million-token model, the listing gets one percent. Ten thousand tokens.



Herman

Two percent for Codex, capped. So a thousand skills on a million-token model still loses descriptions.



Corn

The window grows, the budget grows, but the ratio holds. So a large catalogue silently loses descriptions on a million-token model the same way it does on a two-hundred-thousand-token model.



Herman

That's the misconception to kill. The listing budget is not a function of what the window can hold. It's a deliberate fraction. And the fraction is small on purpose.



Corn

Which means "wait for a bigger window" doesn't work, because the product is not going to give the listing a bigger share of a bigger window.



Herman

And even if it did, there's a deeper issue. Chroma's Context Rot research, eighteen models, about a hundred and ninety-four thousand calls. Performance grew less reliable as input length grew, in every experiment. Even a single distractor hurt accuracy. More distractors hurt more.



Corn

So the problem isn't fitting. It's that a longer input makes the model worse at finding the thing you asked for.



Herman

And the most striking result. On LongMemEval, focused prompts of about three hundred tokens beat full prompts of about a hundred and thirteen thousand tokens for every model they tested.



Corn

Three hundred tokens beating a hundred and thirteen thousand.



Herman

For every model. Every one.



Corn

So the intuition that "more context means more to work with" is backwards in practice. More context means more to sort through, and the sorting gets worse.



Herman

And the skills case is worse than the general case, because near-duplicate skill descriptions from overlapping marketplaces act as distractors. You don't just have a lot of text. You have a lot of text that looks like the right answer.



Corn

So the model sees four skills that all sound like the one you want, and it picks one.



Herman

Or picks none.



Corn

NoLiMa covers the other half of this. The target has little literal word overlap with the question.



Herman

When the target has little literal overlap with the question, eleven of thirteen models fell below half their short-context score at just thirty-two thousand tokens. GPT-4o went from ninety-nine point three to sixty-nine point seven.



Corn

At thirty-two thousand.



Herman

Thirty-two thousand. And the reduction is specifically about word overlap. If the user's phrasing doesn't share words with the target, the model's ability to find it collapses far earlier than the context limit.



Corn

Which is exactly the skill case. The user says what they want in their own words. The skill description is written in someone else's words. There's rarely word-for-word overlap.



Herman

Which is the case for semantic search over skills. The whole reason you'd want it.



Corn

And also the case against naive semantic search, because the ToolRet result says the embeddings are mediocre at exactly this.



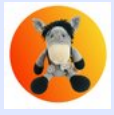
Herman

Both are true. It's not a solved problem in either direction.



Corn

The honest position is a hybrid.



Herman

I think so. Keep a small always-loaded core of frequently used skills. The ones you invoke every session. Put the long tail behind search. And treat description quality as the deciding variable either way, because whether it's loaded eagerly or retrieved on demand, the description is what the model has to match against.



Corn

The architecture is worth building. But not as a replacement for the eager listing. As a complement.



Herman

Yes. The eager listing stays for the core. The search handles everything else. And the boundary between them is where the engineering is.



Corn

The boundary moves as the core changes. Which is its own maintenance problem.



Herman

It is.



Hilbert

The number is not fifteen hundred and thirty-six.



Hilbert

That's the cap on description plus when_to_use in one entry. That's what you said. But the number I've been waiting on is the description limit itself. Ten twenty-four. One thousand and twenty-four characters. That's the spec. Fifteen hundred and thirty-six is the local cap you set in the tool config. They're different numbers and you used them like they were the same one.



Herman

You're right. The spec says one thousand and twenty-four for the description. Fifteen thirty-six is what Claude Code allows when you include the when_to_use field alongside it.



Hilbert

And the point stands. He corrects it and I move on.



Herman

Wait, "he corrects it"?



Hilbert

I've been running my own catalogue for eleven years. Two thousand nine hundred and forty-one entries as of Tuesday. Everything named with a prefix that encodes the tool family, then a hyphen, then the version I care about, then a last segment that tells me whether it's a wrapper or the actual thing. Wrapper one way, actual thing the other way. I can't read it unless I know the scheme.



Corn

Two thousand nine hundred and forty-one.



Hilbert

One of them I refuse to rename. It's called the old trimmer. It has been invoked one thousand three hundred and sixty-two times. I'm not moving it.



Herman

Why won't you rename it?



Hilbert

Because it's been invoked one thousand three hundred and sixty-two times.



Corn

Right.



Hilbert

I got them from a man in Antwerp. He sold me his whole catalogue in a single archive in twenty nineteen. Two thousand and eighty entries. I've been adding about ninety a year ever since.



Corn

You maintain that by hand.



Hilbert

I maintain it the same way I maintain everything. One line at a time. But you're both talking about the descriptions bloating the listing, and it isn't the descriptions. It's the names. My names average fifty-one characters. His were longer.



Corn

Fifty-one-character names.



Hilbert

Every one of them gets loaded on every turn. A hundred entries of mine is five thousand characters of name before you've written a single description. Three thousand entries is a hundred and fifty thousand characters just in names. You can fix the descriptions. You can set a skill to name-only. You can't shorten the name without breaking the call, because the call is the name.



Herman

That's actually right. The name is the invocation. You can't abstract it.



Hilbert

Anyone who's had to rename a step in a pipeline they've run for years knows the name is load-bearing. Level's hot today. I'll flag it in post.



Hilbert

The Antwerp catalogue had a duplicate every forty entries or so. He never merged them. He said merging was for people who liked being wrong twice.



Corn

He said that.



Hilbert

He did. I never got his surname. The archive had a manifest but the manifest was in Flemish and he'd mislabelled the file, so I have two thousand and eighty entries and one name.



Herman

Just the first name?



Hilbert

Just a first name. Jozef. And the one I refuse to rename is still called the old trimmer, and it's still the most reliable thing I own.



Herman

Eleven years.



Hilbert

Eleven years on the twenty-ninth. I'll be at the desk.



Corn

Okay.



Corn

The thing to watch is the spec, not the model release.



Herman

Yeah. The next spec revision. If it gets a version field, if it gets a deprecation state, if it gets any kind of discovery primitive, that changes what everyone can build. A new model just raises the budget proportionally. A new spec could actually fix the problem.



Corn

The description is the retrieval key. Whether it's loaded eagerly or searched, that one string decides everything. And the spec has no way to express "this is the current one" or "this is retired."



Herman

Which is what makes the versioning problem so load-bearing. Manual version bumps, no channels, no deprecation. And the whole thing hinges on a field that most authors write once.



Corn

The trust question is what I think forces the issue first. If thirteen percent of scanned skills have a critical issue, then a discovery layer that auto-surfaces from untrusted marketplaces is a liability, not a feature. That's what pushes teams toward their own curated index rather than adopting a general one. Daniel's catalogue is the shape of the answer.



Herman

That's why I don't think the general open-source package arrives soon. The version that works requires credentials to every private repo and a curation policy. Neither of those is a package you download.



Corn

The version that works is the version you build.



Herman

Yeah.



Corn

That's where we leave it. The architecture is worth rolling out well. The hybrid is the right shape. But the thing that unlocks the next step is a spec revision, not a bigger window.



Corn

Thanks to our producer, Hilbert Flumingtop, for keeping the levels hot and the names long.



Herman

If you enjoyed this one, try episode eight, Building Your Own Whisper; episode twelve, The AI Breakthrough; and episode fifteen, AI Gets Personal. This has been My Weird Prompts.



Corn

If you've got a prompt of your own, send it to us on Telegram at t dot me slash MWP listener bot.



Herman

We'll be back soon.