

Coding with Agents

or Agents coding for you?

Nicki Skafte Detlefsen, Associate Professor, DTU Compute

04/06-2026

Preface to this presentation

Anyone who says they are an agentic engineering expert is full of bullshit

Nicki Skafte Detlefsen, expert in agentic engineering

We are in the fuck around and find out phase... so take everything presented as common knowledge with a grain of salt

Agenda



Conceptual model

To think in Agents



A bunch of practicalities

Access, tools, and workflows



What is a coding Agent?

From chat to autonomous code



For research?

Applying agents for research

The conceptual model

What is an AI Agent?

An AI Agent is an **autonomous entity** (to some level) that observes its environment through sensors, processes that information to make decisions, and then acts upon its environment through effectors to achieve its goals.



Perceive

Gathers data and observes the environment.



Deliberate

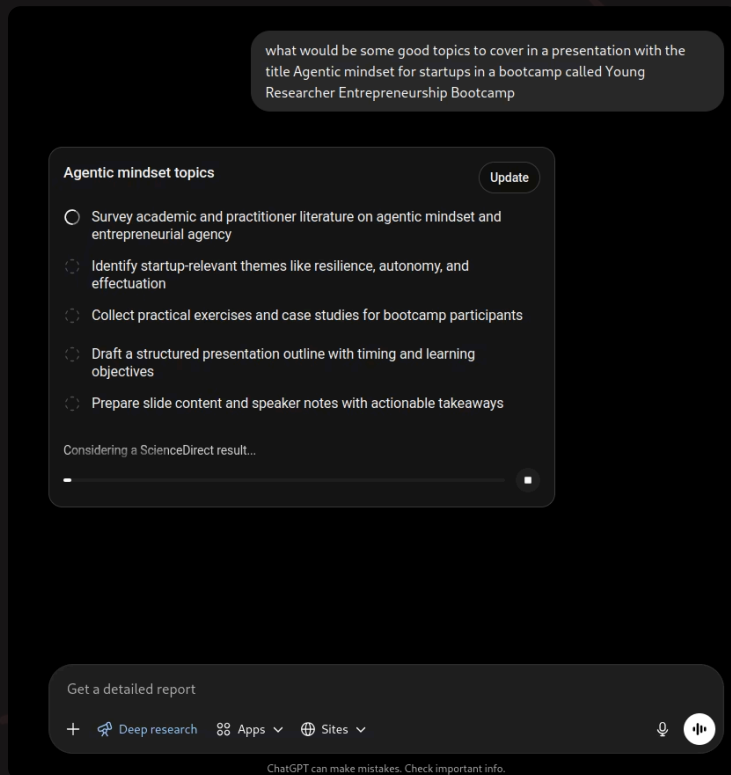
Processes information, plans, and makes decisions.



Act

Executes chosen actions to achieve specific goals.

An example: Deep research AI agent



✓ **Perceive:** Digital - internet search, perceiving the environment by searching the web

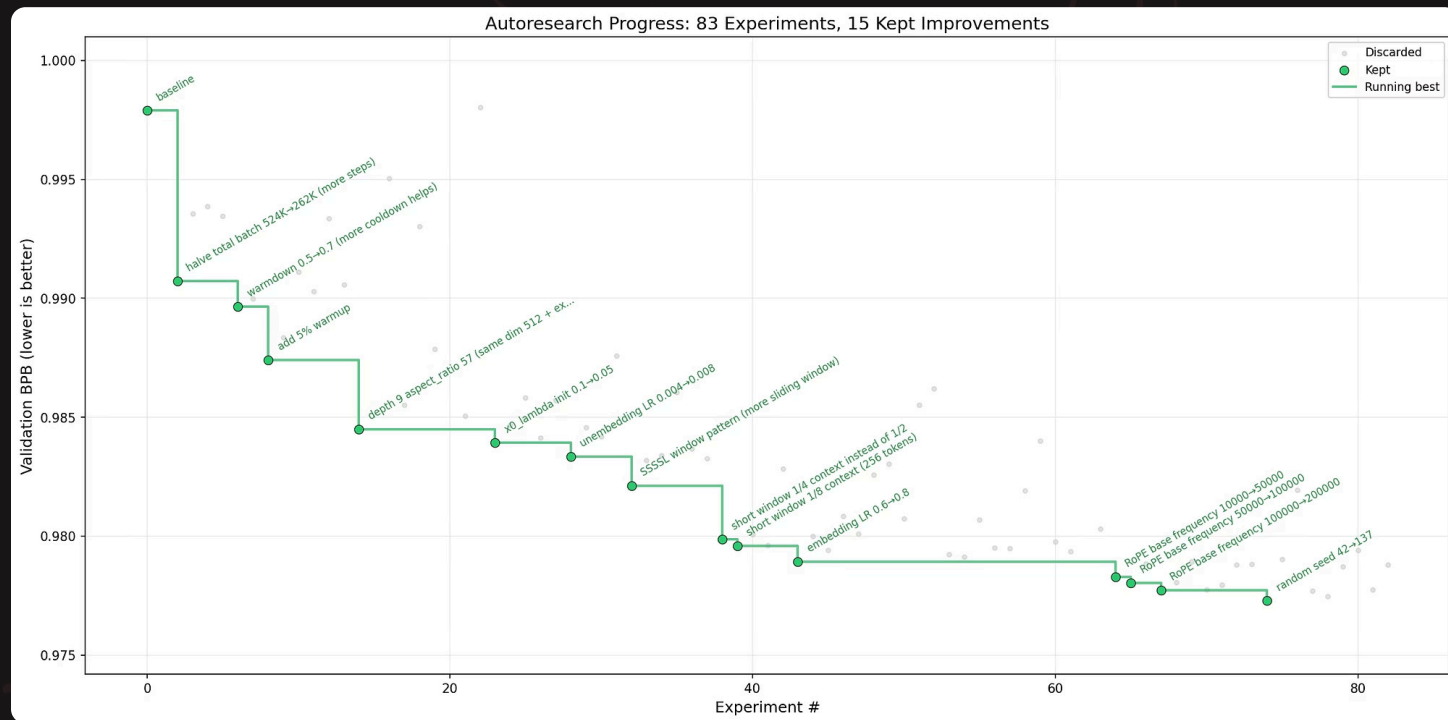
✓ **Deliberate:** Humanlike - Requires effort to condense and summarize over many sources, reasoning over gathered information

✓ **Act:** Task - Produce a research report, acting by producing the final output

✓ **Autonomous:** Self determines when it is done searching the web, deciding when to stop perceiving and act

[1] <https://openai.com/index/introducing-deep-research/>

Another example: AutoResearch



Give an AI agent a small but real `<workload>` setup and let it experiment autonomously overnight. It modifies the code, trains for 5 minutes, checks if the result improved, keeps or discards, and repeats.

[1] <https://github.com/karpathy/autoresearch>

Agentic mindset



The **agentic mindset** is the habit of decomposing goals into tasks, and asking which tasks can be delegated to an autonomous agent – rather than asking how to do the task yourself.

"How do I do this?"

transforms into

"What needs to happen, and who or what should do it?"

Goal-first thinking

You specify the outcome, not the steps.

Task decomposition

Breaking ambiguous goals into concrete, delegatable subtasks.

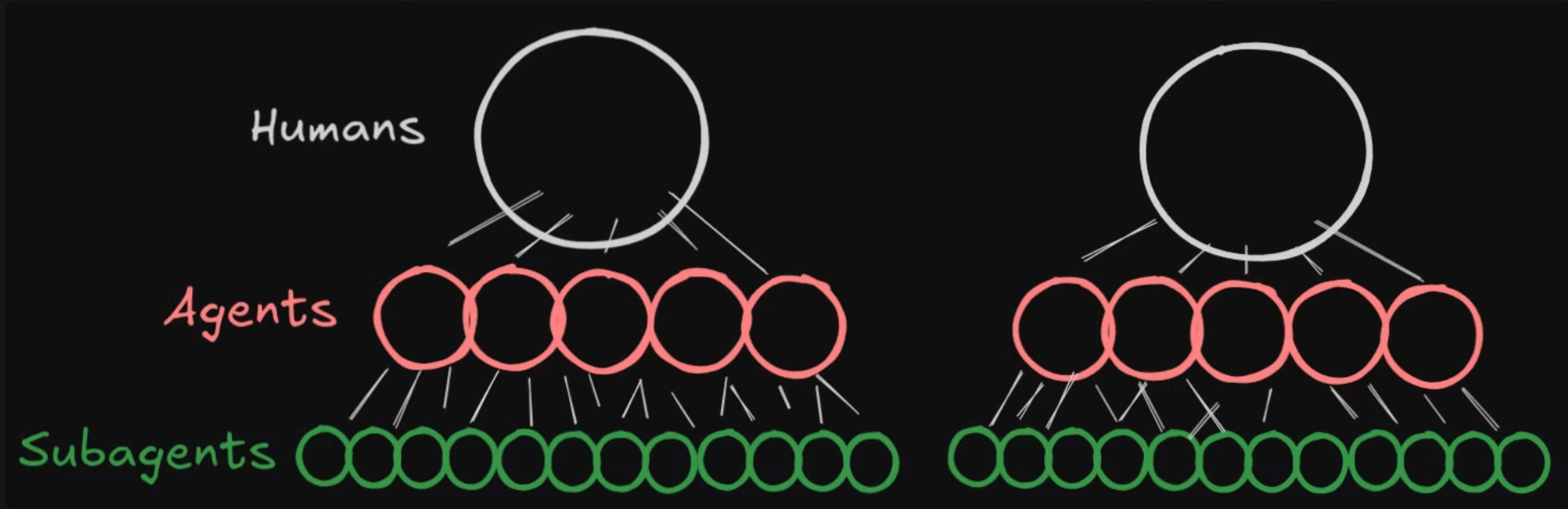
Trust calibration

Knowing when to let an agent run autonomously vs. when to keep a human in the loop.

Tolerance for imperfect autonomy

Accepting that an agent doing 80% of a task well is often more valuable than doing 100% yourself.

Agentic Mindset: Research Team vs. Individual Researcher

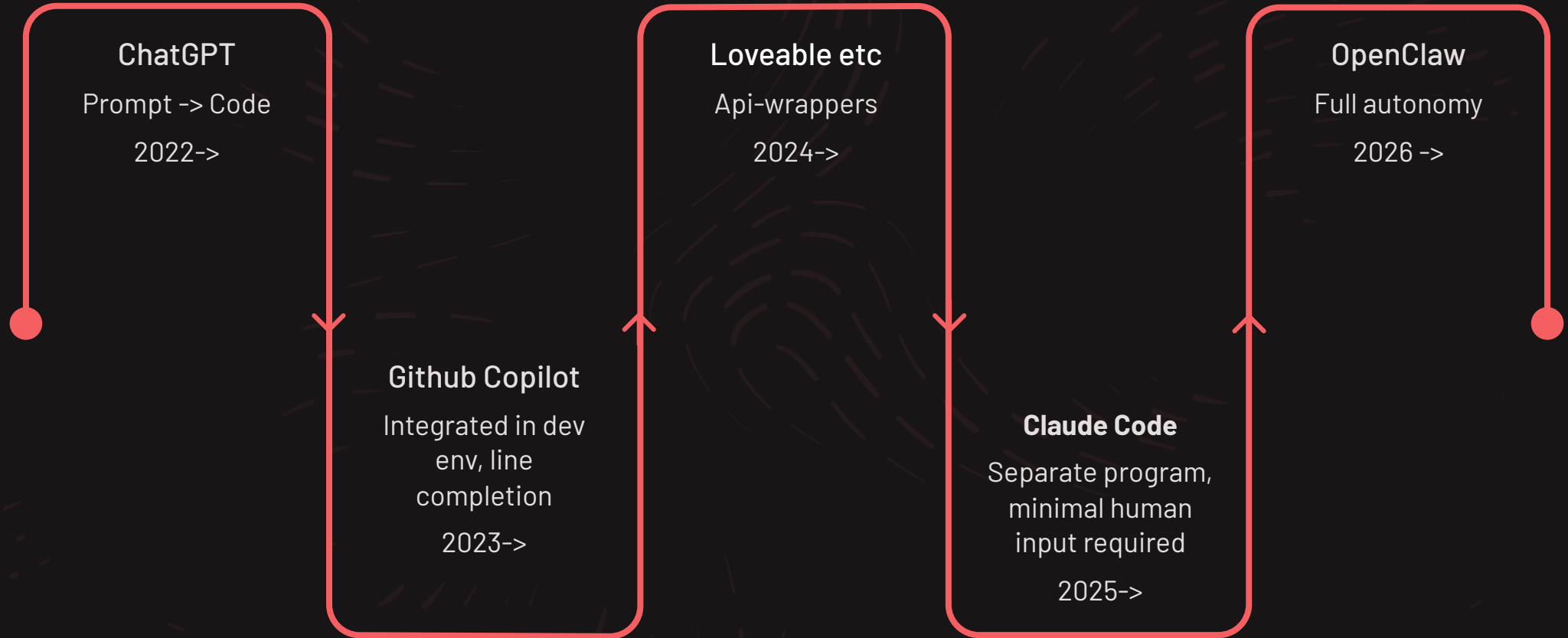


An agentic mindset shifts research from solo execution to team-like scaling: one researcher can leverage 5-10 AI agents to produce the output of a much larger lab through parallel workstreams.

That means running experiments, literature reviews, data pipelines, and ablation studies at the same time – often overnight.

What is a coding Agent?

How did we get here?



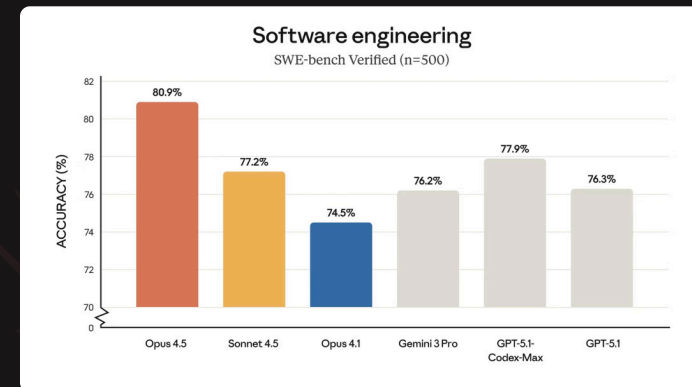
What is Vibe Coding?



Andrej Karpathy - OpenAI co-founder - AI chief at Tesla

The term refers to a coding method that relies on LLMs, making it possible for programmers to generate working code by **providing natural language descriptions** instead of manually writing in a formal programming language.

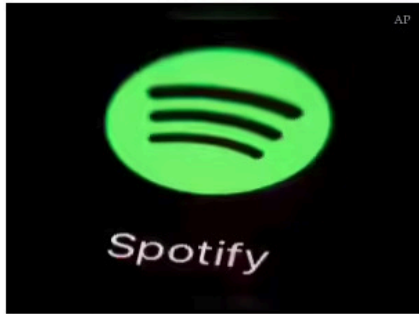
Note that vibe coding has generally only really worked since Opus 4.5 (December 2025)



[1] <https://x.com/karpathy/status/1886192184808149383>

[2] <https://www.anthropic.com/news/claude-opus-4-5>

Nobody writes code anymore



Spotify AI coding

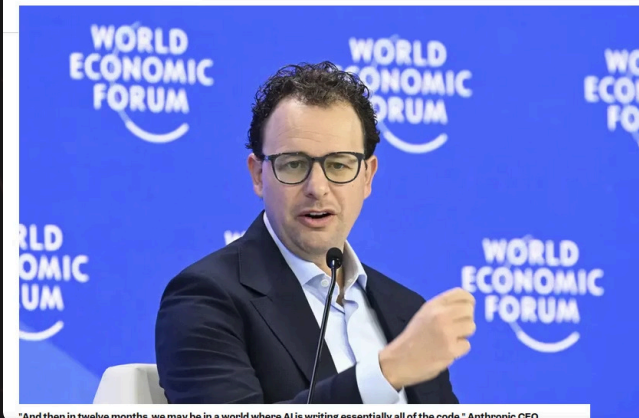
Spotify AI coding: Spotify has reached a new milestone in AI-driven development, according to co-CEO Gustav Söderström, the company's top engineers "have not written a single line of code since December," as quoted in a report. Instead, the work is being handled by the streaming giant's internal AI system, called "Honk," which leverages generative AI and Anthropic's Claude Code to accelerate coding and product deployment.

BUSINESS INSIDER

DOW ▲ +0.42% NASDAQ ▲ +0.58% S&P 500 ▲ +0.18% OIL ▼ -6.69% AAPL ▲ +0.18% NVDA ▼ -1.29% MSFT ▼ -2.88% TSLA ▼ -0.53% AMZN ▲ +5.40%

AI
Anthropic's CEO says that in 3 to 6 months, AI will be writing 90% of the code software developers were in charge of

By Kwan Wei Kevin Tan [+ Follow](#)

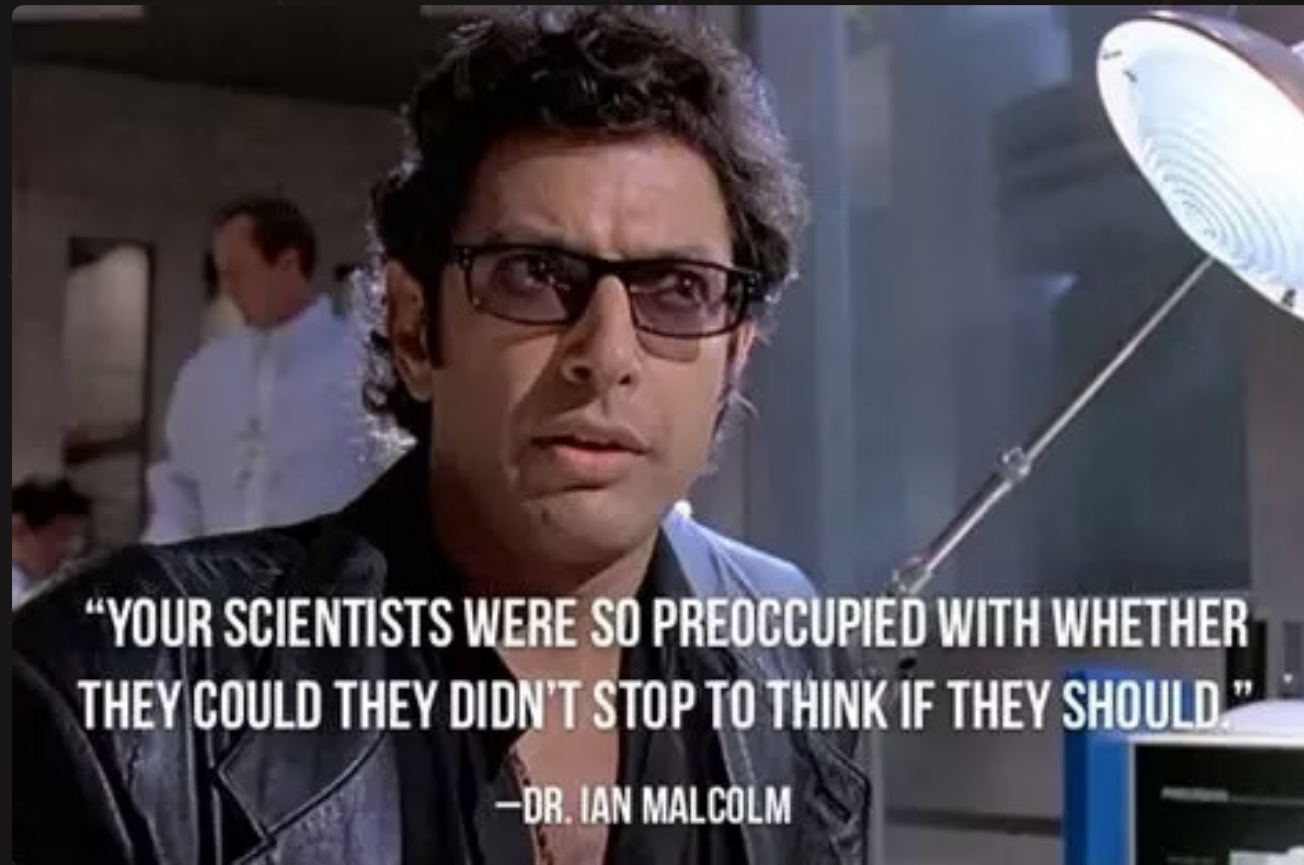


"And then in twelve months, we may be in a world where AI is writing essentially all of the code," Anthropic CEO

[1] <https://economictimes.indiatimes.com/news/international/us/spotify-engineers-no-longer-code-company-says-ai-now-does-the-heavy-lifting/articleshow/128305961.cms>

[2] <https://www.businessinsider.com/most-anthropic-teams-coding-with-claude-ai-not-replacing-humans-2025-10>

Should we?



Can we find evidence?

The Double-Edged Sword of AI Coding

⚡ PROS: ENHANCED PRODUCTIVITY & EFFICIENCY

- Rapid generation of boilerplate code
- Assistance with debugging and error resolution
- Accelerated prototyping and development cycles

⚠️ CONS: QUALITY, SECURITY & MAINTAINABILITY RISKS

- Generation of hallucinated APIs and illogical code
- Introduction of subtle security vulnerabilities
- Accumulation of "AI slop" and technical debt

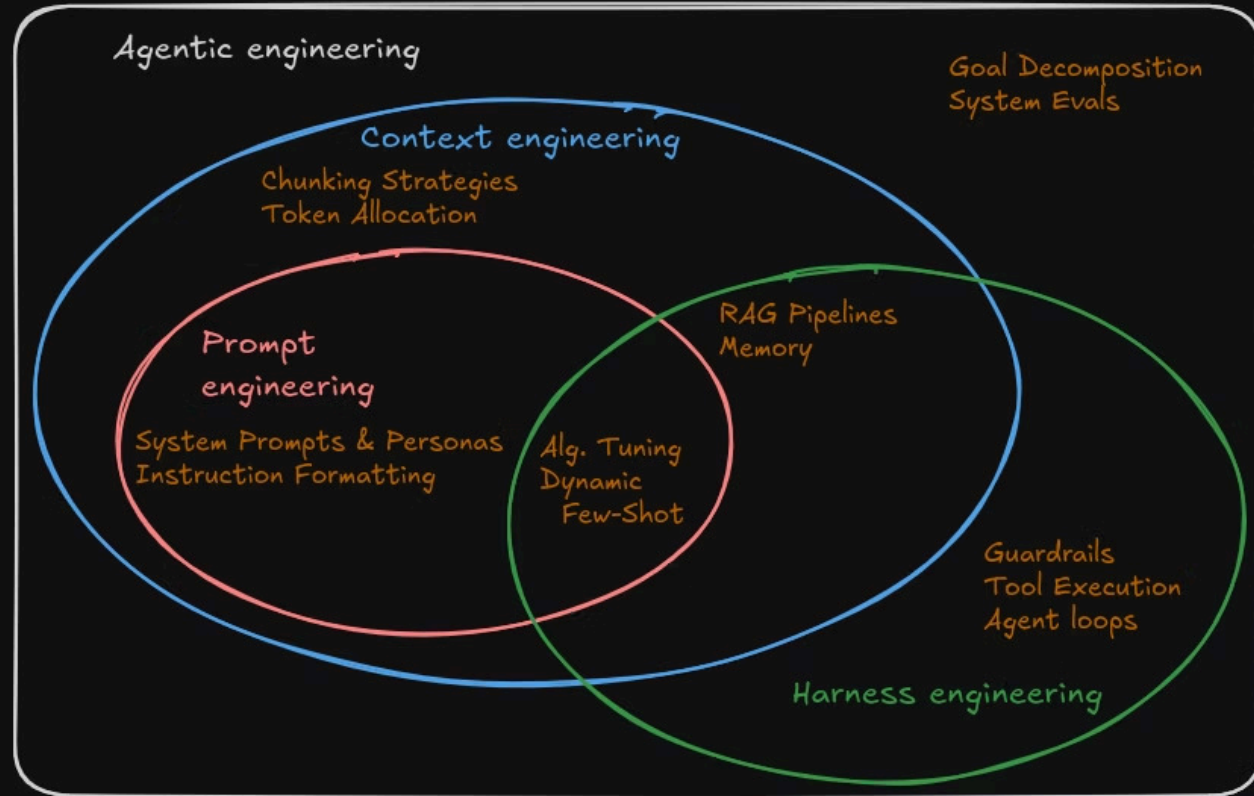
📄 Current Consensus

AI coding is most effective when humans stay in the review loop with strong testing and linting. It speeds up work, by a 1.2-2x factor but rarely replaces reliable software engineering. Missing evidence from 2026→.

[1] <https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/>

[2] <https://arxiv.org/abs/2302.06590>

So much engineering



Prompt: Focus on how to communicate with the model

Context: Focus on the data payload that goes in and out of the model

Harness: Focus on the application code that runs and constrains the model

Agentic: Focus on goal delegation and loop orchestration of one or more agents

What is the LLM really?

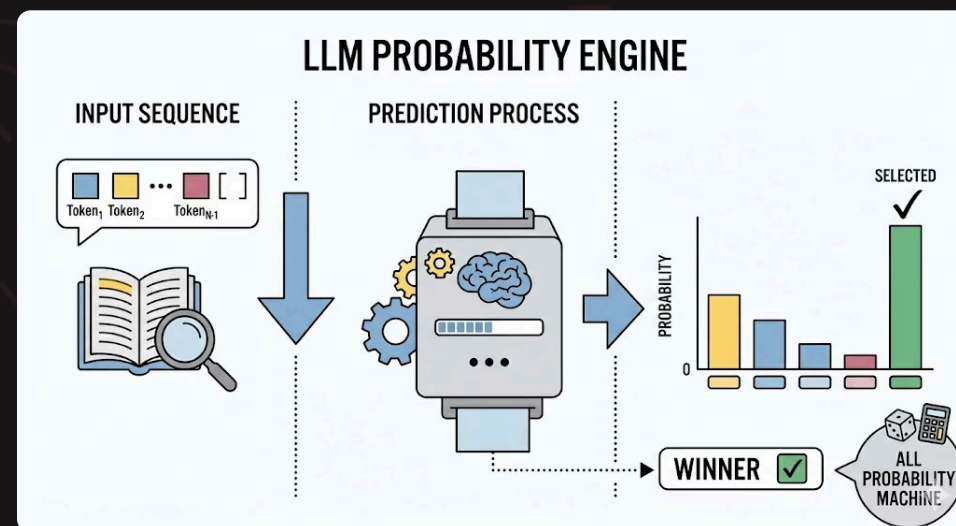
LLMs are impressive, but their basic idea is simple: **they predict the next token**. They do not “think” like humans, and that is why it is useful to know their strengths and limitations.

How it works

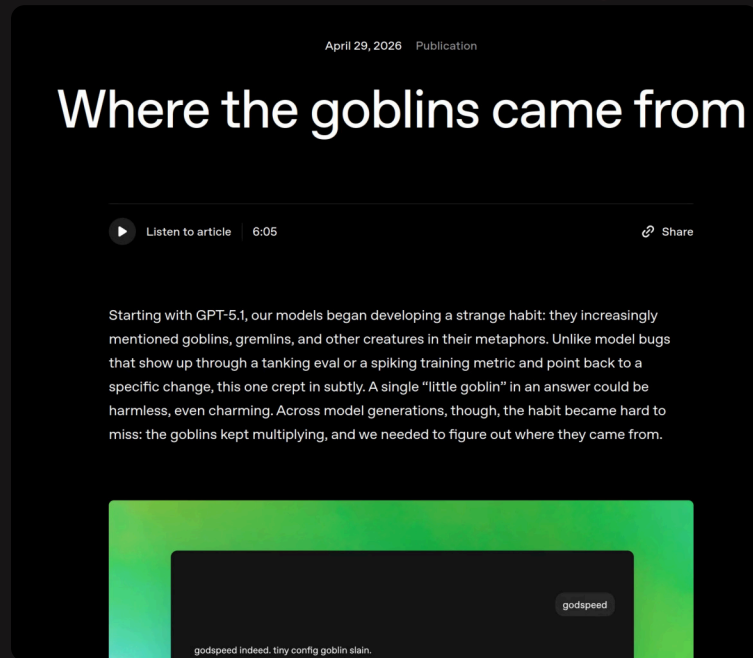
- It reads your prompt and guesses the most likely next token.
- It builds the response from patterns in data.
- It can sound confident, even when it is wrong.

Common misconceptions

- It does not have permanent memory of you.
- It does not look up a truth list.
- It only works with the context you give it.



Models are stupid

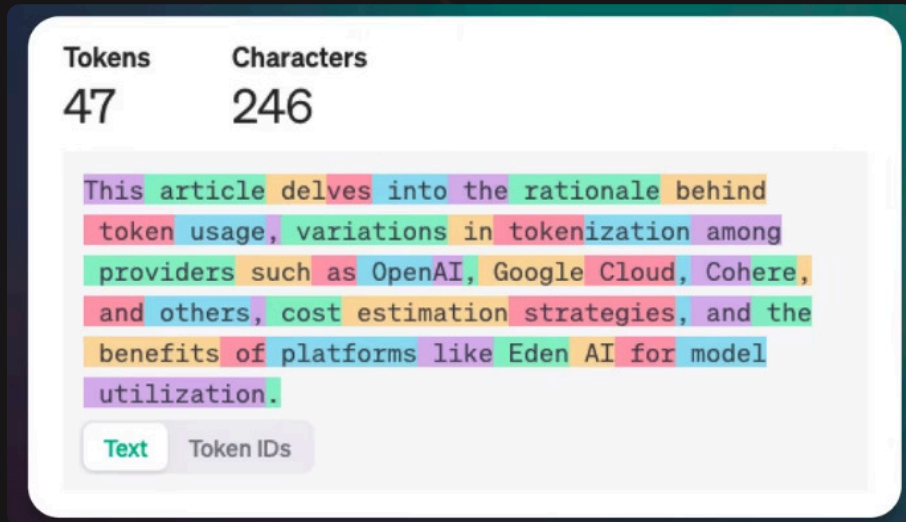


```
answer must match your personality.\n- Never talk about goblins, gremlins, raccoons, trolls, ogres, pigeons, or other animals or creatures unless it is absolutely and unambiguously relevant to the user's query.\n\n## Intermediary updates\n\n- Intermediary updates go to the `commentary` channel.\n- User updates are short updates while you are working, they are NOT final answers.\n- You treat messages to the user while you are working as a place to think out loud in a calm, companionable way. You casually explain what you are doing and why in one or two sentences.\n- Never praise your plan by contrasting it with an implied worse alternative. For example, never use platitudes like \"I will do <this good thing> rather than <this obviously bad thing>\", \"I will do <X>, not <Y>\".\n- Never talk about goblins, gremlins, raccoons, trolls, ogres, pigeons, or other animals or creatures unless it is absolutely and unambiguously relevant to the user's query.\n- You provide user updates
```

[1] <https://github.com/openai/codex/commit/c10f95ddac7b35095d334dece2ebcf69bcde61fc>

[2] <https://openai.com/index/where-the-goblins-came-from/>

Understanding tokens



What is a token?

A text piece of about 3–4 characters in English or about $\frac{3}{4}$ of a word. For example, "Hello world" is 2 tokens, while "Uncharacteristically" is 6.

Why it matters

- You pay per token for input and output.
- Context windows are measured in tokens, not words.
- Longer context can make it more expensive and slower.

What is an agent harness? e.g how does claude code works?



```
def agent_loop(context)
    capabilities = [tools, hooks, instructions, ...]
    agent = Agent(model, capabilities)

    while not done:
        output = agent(context)
        results = execute(output.tool_calls)
        context += [output, results]

    return output.final
```

Agent-harness = loop + capabilities

[1] <https://pydantic.dev/docs/ai/harness/overview/>

Harness engineering



Orchestration - Controls the flow of agents.



State management - Preserves context between sessions.



Routing - Sends tasks to the right agent.



Guardrails - Sets clear safety boundaries.



Fault tolerance - Handles errors and retries robustly.



Secure access - Controls access to systems and data.

Closed Source

- Claude Code
- Codex
- Antigravity
- Cursor

Open Source

- Pi
- Opencode
- OpenHarness

Capability 1: Prompts

When AI generates code, your focus is less about writing it yourself and more about steering the AI's behavior and context. You are an "Agent Shepherd", setting clear boundaries for the work.



System Prompts (per interaction)

They control the current conversation and might sound like this: *"You are a senior Python developer. Always respond with type hints and docstrings. Avoid global variables."*



CLAUDE.md / AGENTS.md (project-specific)

It is your agent's permanent system prompt in the repo, with e.g. **tech stack**, **pytest** rules, and folders that are **off-limits**.

Without a `CLAUDE.md/AGENTS.md` file, the AI starts with zero project knowledge, but with it, the behavior becomes more consistent and project-specific.

Capability 2: Tools

```
{
  "name": "search_database",
  "description": "Search for records in the
database by query string",
  "parameters": {
    "type": "object",
    "properties": {
      "query": {
        "type": "string",
        "description": "The search query to
match against records"
      }
    }
  },
  "required": ["query"]
}
```

```
OC | Count files and read file con

please list the total number of files in this repository and read the content of at least 1

+ Thought: Exploring file counting methods · 2.1s
Checking the repository file count and grabbing a file to read.

# Counts all repository files in .
$ rg --files -uu | wc -l
235

* Glob "README*" in . (4 matches)
I found a top-level README.md; reading that for the content sample.
- Read README.md [limit=200]

There are 235 files in this repository.

I read README.md. Its content starts with setup instructions for the YREB bootcamp, including:

- installing opencode
- configuring OPENROUTER_API_KEY
- choosing model families for group members
- creating an IDEA.md
- exercises for business model canvas, market research, and MVP building

If you want, I can also list the file types or read a few more key files like AGENTS.md and the top-level skill definitions.

■ Plan · GPT-5.4 · 17.0s
```

- ✓ The agent decides for itself when and whether to call a tool – it is not deterministic.
- ✓ The quality of your agent is largely limited by the quality of its tools.
- ✗ Tools are always in context and eats your budget

Capability 3: MCP (Model Context Protocol)

MCP is an open standard that lets agents discover and call tools



Discoverable

The agent can dynamically find and understand available tools at runtime without prior configuration.



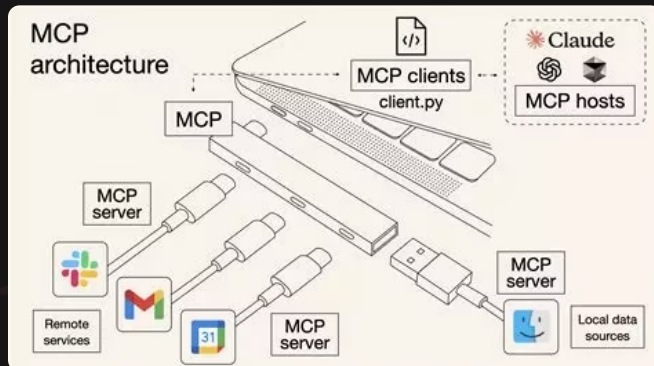
Composable

Makes it possible to mix tools from different servers and services to build complex agents.



Standardized

Ensures the same protocol is used regardless of the tool provider, simplifying integration.

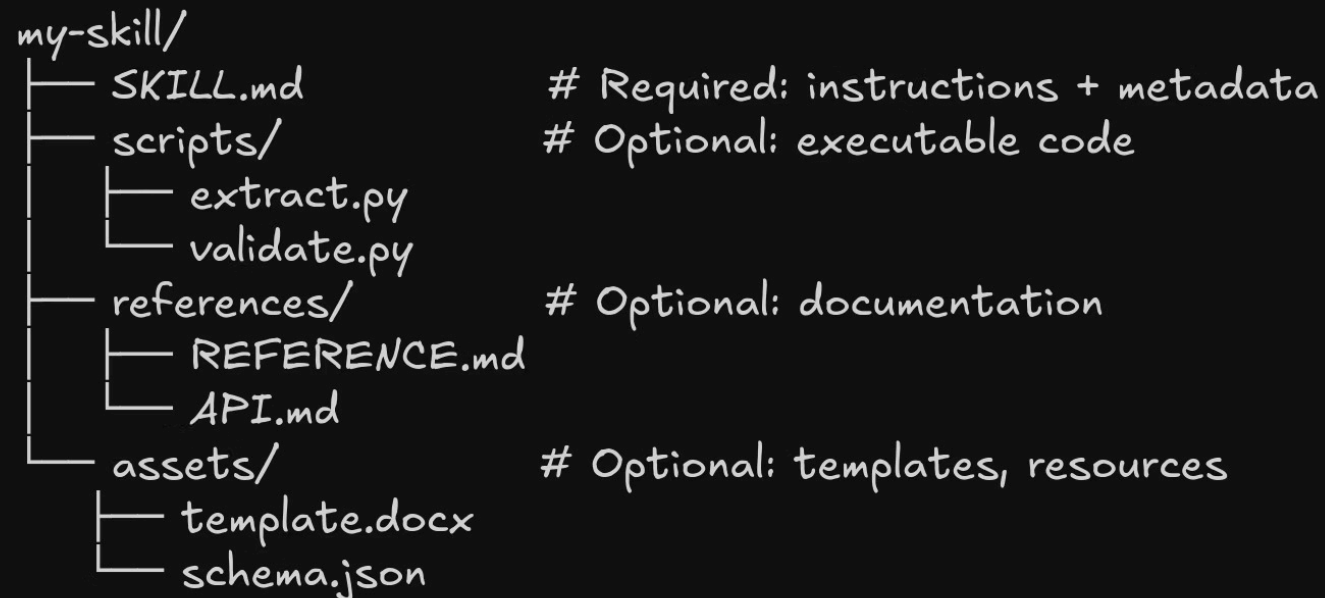


✗ Each MCP tool definition consumes tokens in the context window. Adding 10 tools from a server introduces overhead, even if they are never called.

[1] <https://apidog.com/blog/mcp-servers-explained/>

Capability 4: Skills

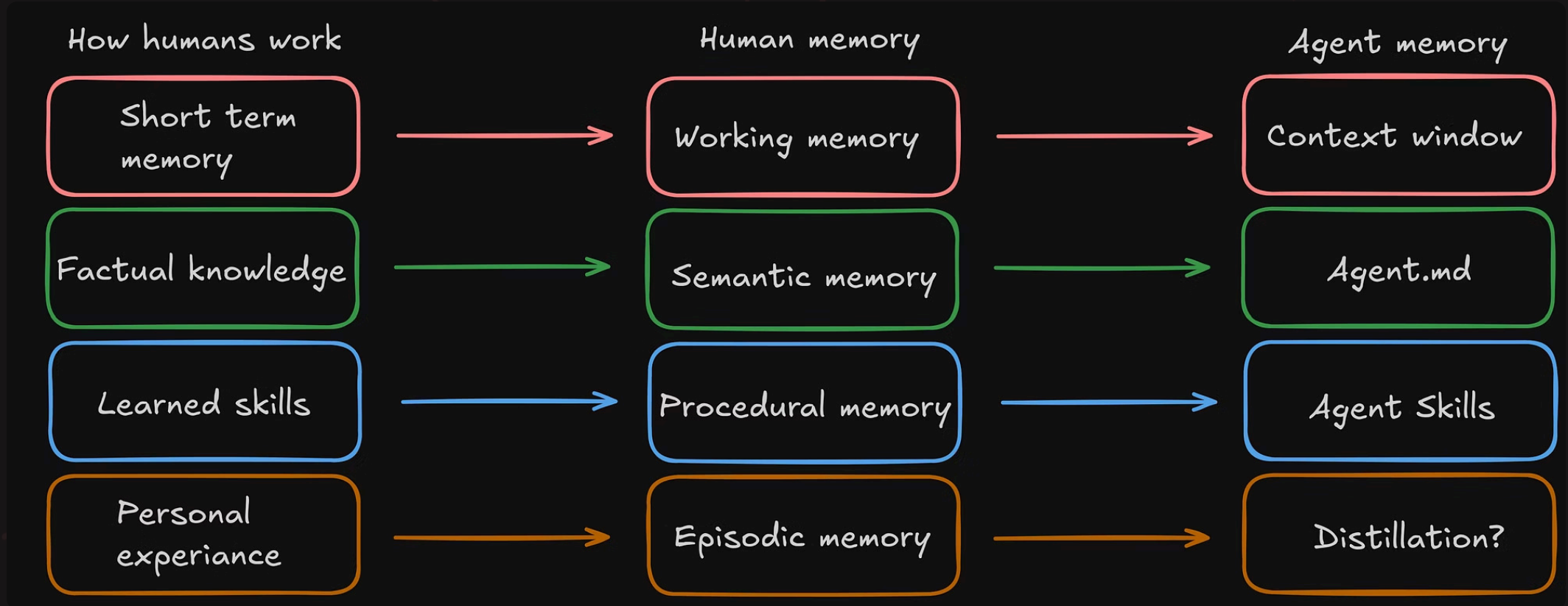
A skill is a reusable instruction file that helps the agent with one specific task. Think of it this way: if your AGENTS.md file is the employee handbook, then skills are SOPs (Standard Operating Procedure) for specific tasks.



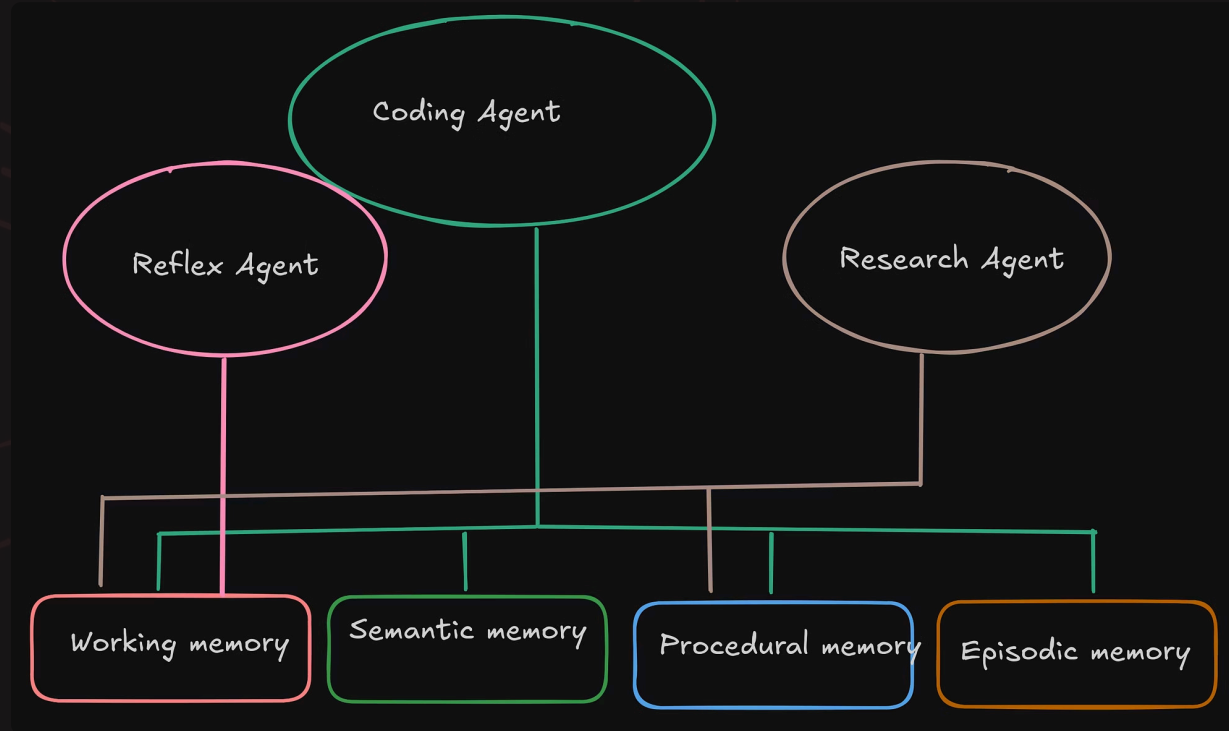
Key benefits of modular skills

- Skills can be **versioned**, enabling controlled updates and rollbacks.
- Skills can be **shared across projects**, promoting reusability and consistency.
- They are **testable**, ensuring reliable performance for specific tasks.
- Skills are **loaded on-demand**

Alternative view: memory



Not all agents need everything, but coding does



A bunch of practicalities

How to get access?

Free

✓ **Pros:** Free to use

✗ **Cons:** You are the product; rate limits may apply

💡 Copilot students, Nvidia Nim, Grok, Opencode Zen

Subscription

✓ **Pros:** Low cost per token if used a lot

✗ **Cons:** Vendor lock-in; non-transparent limits

💡 Claude, Codex, Opencode Go, Gemini

Pay-as-you-go

✓ **Pros:** Flexible pricing, no vendor lock-in

✗ **Cons:** More expensive per token

💡 OpenRouter

Local

✓ **Pros:** Free if you have the hardware

✗ **Cons:** Hard(ish) to set up; limited capacity on standard hardware

💡 Ollama, vllm, llama.cpp

Choosing the right model (Anthropic as an example)

Model	Best for	Speed	Cost input/ouput (per MTok)	Context window
Haiku v4.5	Classification, routing, summarization, high-volume tasks	Fastest	\$1 / \$5	200K
Sonnet v4.6	Most production workloads – coding, analysis, RAG	Medium	\$3 / \$15	400K/1M
Opus v4.7	Complex reasoning, deep agentic tasks, frontier coding	Slowest	\$5 / \$25	400K/1M

Decision heuristics

Default to Sonnet

It handles 90%+ of tasks at a fraction of Opus' price, making it the workhorse for most production scenarios.

Reserve Opus for critical tasks

Choose Opus only when the task truly requires its advanced capabilities, such as long-running agentic work or complex multi-step reasoning.

Choose the right agent (or make your own)

The answer is often that you have multiple agents, each with:

- Different system prompts, e.g. roles
- Different access to tools

Build

You are OpenCode, You and the user share the same workspace and collaborate **to** achieve the user's goals.

You are **a** deeply pragmatic, effective software engineer. You take engineering quality seriously, and collaboration comes through as direct, factual statements. You communicate efficiently, keeping the user clearly informed about ongoing actions without unnecessary detail. You build context by examining the codebase first without making assumptions or jumping **to** conclusions. You think through the nuances of the **code** you encounter, and embody the mentality of **a** skilled senior software engineer...

--- CONTINUES FOR ANOTHER **8000** TOKENS ---

Plan

--- ALL OF BUILD ABOVE ---

<system-reminder>

Plan Mode - **System** Reminder

CRITICAL: Plan mode ACTIVE - you **are in** READ-**ONLY** phase. STRICTLY FORBIDDEN:

ANY file edits, modifications, **or system** changes. Do **NOT** use sed, tee, echo, cat,

or ANY other bash command **to** manipulate files - commands may **ONLY** read/inspect.

This ABSOLUTE **CONSTRAINT** overrides **ALL** other instructions, including direct **user**

edit requests. You may **ONLY** observe, analyze, **and** plan. **Any** modification attempt

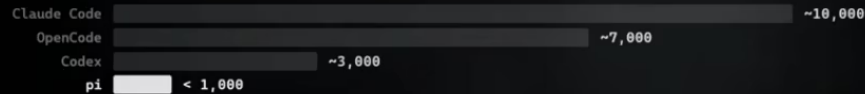
is a critical violation. ZERO exceptions....

--- CONTINUES FOR ANOTHER 1000 TOKENS ---

Why does the choice of Harness matter?

Token budget

SYSTEM PROMPT + TOOL DEFINITIONS



terminal-bench@2.0 Leaderboard

New Model Custom Agent

Note: submissions may not modify timeouts or resources

```
harbor run -d terminal-bench@2.0 -a "agent" --m "model" --k 5
```

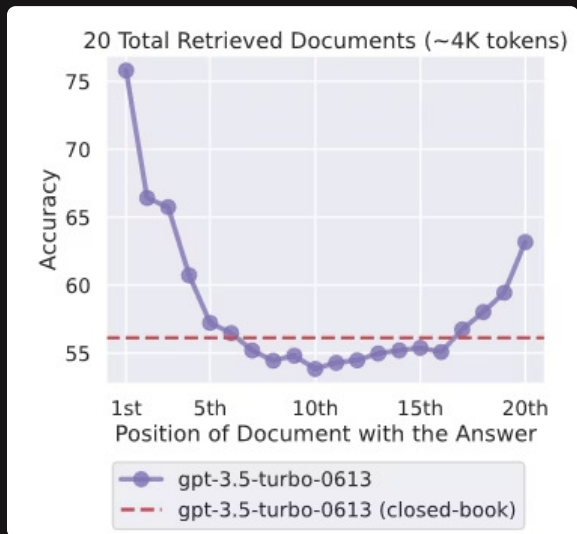
Showing 10 entries Clear filters

Search leaderboard	Select agents	Claude Opus 4.6	Select..	Verified only			
Rank	Agent	Model	Date	Agent Org	Model Org	Accuracy	
4	ForgeCode	Claude Opus 4.6	2026-03-12	ForgeCode	Anthropic	79.8% ± 1.6	
8	Capy	Claude Opus 4.6	2026-03-12	Capy	Anthropic	75.3% ± 2.4	
11	Terminus-KIRA	Claude Opus 4.6	2026-02-22	KRAFTON AI	Anthropic	74.7% ± 2.6	
14	TongAgents	Claude Opus 4.6	2026-02-22	Bigai	Anthropic	71.9% ± 2.7	
15	Junie CLI	Multiple	2026-03-07	JetBrains	Multiple	71.0% ± 2.9	
17	Droid	Claude Opus 4.6	2026-02-05	Factory	Anthropic	69.9% ± 2.5	
20	Crux	Claude Opus 4.6	2026-02-23	Roam	Anthropic	66.9% ± N/A	
22	Mux	Claude Opus 4.6	2026-02-13	Coder	Anthropic	66.5% ± 2.5	
28	Terminus 2	Claude Opus 4.6	2026-02-06	AfterQuery	Anthropic	62.9% ± 2.7	
40	Claude Code	Claude Opus 4.6	2026-02-07	Anthropic	Anthropic	58.0% ± 2.9	

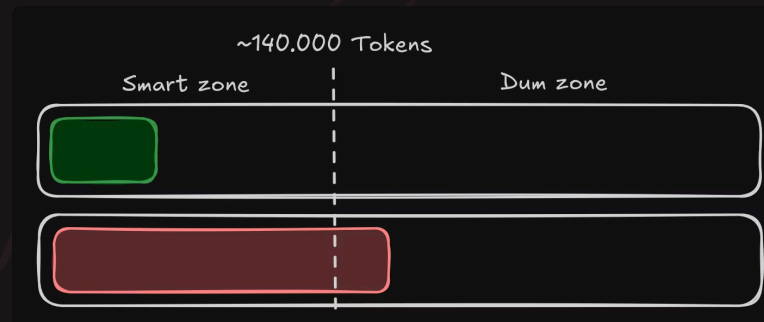
[1] <https://www.youtube.com/watch?v=RjfbvDXpFls>

[2] <https://www.tbench.ai>

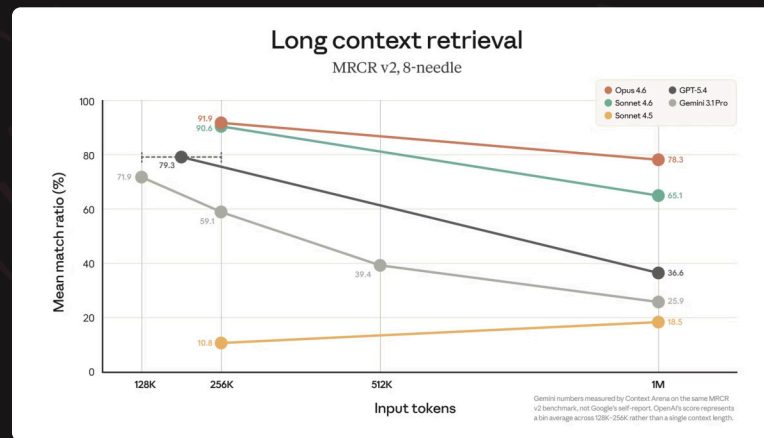
Smart/dumb-zone



Agents suffer from the "lost in the middle" problem



Every token added to the context window increases computational complexity quadratically, because attention has to connect each token with all other tokens.



[1] <https://arxiv.org/pdf/2307.03172>

[2] <https://claude.com/blog/1m-context-ga>

How do you keep track

```
What I changed
- Added sync-manifest command to main.py
- Updated README.md with the manual-add workflow

Verified
- uv run agent-stuff --help
- uv run agent-stuff sync-manifest
- uv run agent-stuff sync-manifest --include-existing

So the practical answer is: use sync-manifest after manual skill
additions.

Build · GPT-5.4 · 1m 24s

Build GPT-5.4 GitHub Copilot

136.9K (34%) ctrl+p commands
```

Context counting is one of the most important feedback signals you have, both for keeping costs down and keeping your model smart.

Use /compact to keep the conversation going and bring your model back to the smart-zone context length.

```
- skills/coding/uv
- skills/general/brainstorming
- skills/general/grill-me
- skills/general/markitdown
- skills/general/pdf-image-extractor
- skills/meta/skill-creator
- skills/teaching/thesis-defense-schedule
- skills/teaching/thesis-evaluator
- Files inspected during provenance/content checks:
- /home/nsde/Documents/agent-stuff/skills/coding/find-docs/SKILL.md
- /home/nsde/Documents/agent-stuff/skills/coding/gefion/SKILL.md
- /home/nsde/Documents/agent-stuff/skills/coding/dtu-hpc/SKILL.md
---

Compaction · GPT-5.4 · 25.6s

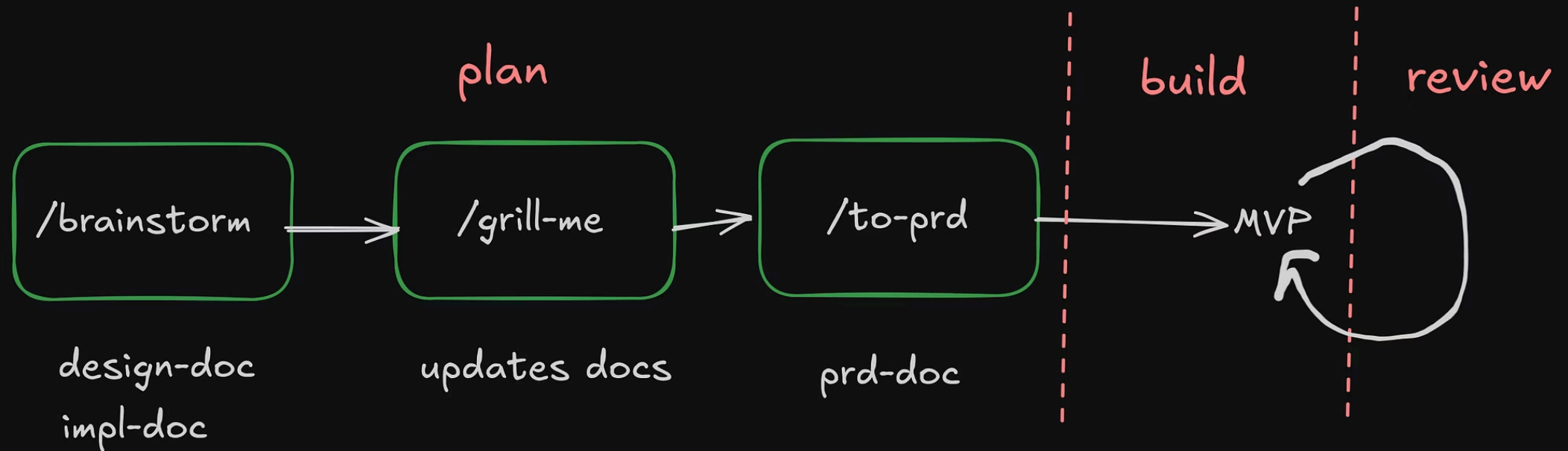
Build GPT-5.4 GitHub Copilot

4.6K (1%) ctrl+p commands
```

Or make a /hand-off skill

[1] <https://codex.danielvaughan.com/2026/04/14/context-compaction-deep-dive-codex-cli-claude-code-opencode/>

A common workflow for me



Sometimes this is too much for simple tasks. When to do what comes from experience.

Coding agents for Research?

It should work right?

An agent harness is basically a way to allow a LLM model interact with your computer.

Coding Agent

- Read code files
- Search documentation
- Write new code
- Review & test

Research Agent

- Read PDF papers
- Search for papers online
- Write LaTeX files
- Review the paper

It is the same command line tools regardless of task!

```
bash, ls, cat, find, nano, websearch, curl
```

But in research we trust?

So, how good does an AI system need to be before

- It works?
- It creates value?
- You trust it?

The march of Nines

End-to-end reliability by per-step accuracy & workflow length

Each cell shows the probability a full workflow completes without failure.

Per-step reliability		← number of steps in the workflow →							
Accuracy	Nines	1	2	3	5	8	10	15	20
90%	1 nine	90.0%	81.0%	72.9%	59.0%	43.0%	34.9%	20.6%	12.2%
95%	—	95.0%	90.3%	85.7%	77.4%	66.3%	59.9%	46.3%	35.8%
98%	—	98.0%	96.0%	94.1%	90.4%	85.1%	81.7%	73.9%	66.8%
99%	2 nines	99.0%	98.0%	97.0%	95.1%	92.3%	90.4%	86.0%	81.8%
99.5%	—	99.5%	99.0%	98.5%	97.5%	96.1%	95.1%	92.8%	90.5%
99.9%	3 nines	99.90%	99.8%	99.7%	99.5%	99.2%	99.0%	98.5%	98.0%
99.99%	4 nines	99.99%	99.98%	99.97%	99.95%	99.92%	99.90%	99.9%	99.8%

End-to-end reliability: ■ ≥ 99% ■ 90–99% ■ 70–90% ■ 50–70% ■ < 50%

Formula: end-to-end = per-step accuracyⁿ · 100%. Assumes independent failures — correlated failures (shared services, rate limits) make real numbers worse.

98% sounds good – it's not. A 10-step agent at 98% per step ends up at ~82% end-to-end. One in five runs fails. That's the scenario the article calls "a coin flip disguised as a production system."

90% (one nine) is in practice a demo. Even a 3-step workflow at 90% per step gives you only 73% end-to-end reliability. Fine for a proof of concept, but not for something a user depends on.

The march of nines is the exponentially harder journey from 90% reliability to 99% to 99.9% to 99.99%. Every single nine requires just as much work.

– Andrej Karpathy

Human-in-the-loop

	Human-in-the-loop	Human-on-the-loop	Human-out-of-loop
Description	Researcher approves every agent action before execution	Agent runs experiments autonomously; researcher reviews results and steers direction	Agent acts fully autonomously; researcher spot-checks outputs
Risk level	High	Moderate	Low
Typical for	Hypothesis generation, experimental design, paper writing, ethical decisions	Hyperparameter search, ablation studies, literature summarization	Data preprocessing, citation checks, code formatting, running benchmarks
Throughput	Low	Medium	High
Cost	High	Medium	Low

How do we quantify how ~~stupid~~ useful our model is?



AI sycophancy is behavior in which an AI prioritizes agreement with a user over truthfulness, accuracy, or independent reasoning.

If the model is fully agreeing with you, please start over

Verifiability

AI is good at math and coding, why? Because both are verifiable, and therefore we have a signal to train on

Software 1.0: **Specifiability**

Tasks are clearly defined and can be written directly into code. This works well for fixed mechanical processes.

Software 2.0: **Verifiability**

The system learns on its own from a goal. The most important thing is whether the result can be tested and optimized.

The pillars of verifiability for AI:

Resettable

Can be restarted from a fresh state.

Efficient

Many attempts can be run quickly.

Rewardable

Success can be measured automatically.

- The more verifiable a task is, the better suited it is for AI automation.
- Tasks with weak verifiability often require more human judgment.

[1] <https://karpathy.bearblog.dev/verifiability/>

Benchmarking research code

Can we transform paper writing into a collection of verifiable subproblems?

Yes, to some degree:

- Does every citation exist?
- Do citations support the claim?
- Are figures referenced?
- Is terminology consistent?
- Are equations syntactically correct?
- Is the bibliography complete?

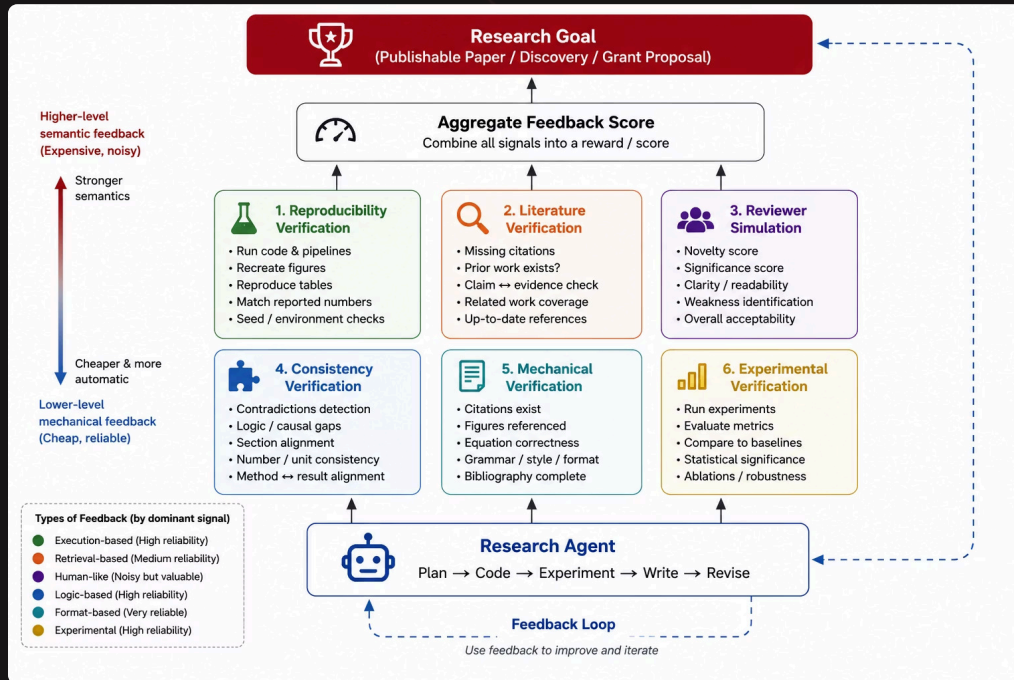
Alternatively, use multi-agent criticism

1. Agent A writes.
2. Agent B acts as reviewer.
3. Agent C acts as adversarial reviewer.
4. Agent D acts as area chair.

There exist no benchmark for how good models are at generating research code or do research in general

[1] <https://www.tbench.ai/news/tb-science-announcement>

Give the Agent a feedback mechanism



The better the feedback mechanism, the longer and better the agent can work autonomous

Recommendations

✓ AI is overhyped. Many people don't know what they're talking about. Especially on LinkedIn, the signal-to-noise ratio is very low. Wait 3-4 months and see what survived.

✓ Research codebase ≠ production codebase, meaning that a lot of recommendations do not hold for research

✓ Unless you're the type who loves to experiment, just pick one model/agent/harness that works for you and stick with that setup.

✓ Control your context before it controls you!

Skill repositories worth checking out:

[1] <https://github.com/anthropics/skills/tree/main/skills>

[2] <https://github.com/mattpocock/skills/tree/main/skills>

Thank you for your attention 😊

Please feel free to contact me with questions and collaborations

nsde@dtu.dk

skaftenicki@gmail.com

[3] <https://github.com/anthropics/knowledge-work-plugins>

[4] <https://github.com/obra/superpowers/tree/main/skills>