Before we begin: open the workshop

Browse the saved conversations

Import the saved conversations
into Agent Canvas.
No model key required.

Want to run the exercises?

Follow the model setup instructions.
Running fresh attempts is optional.

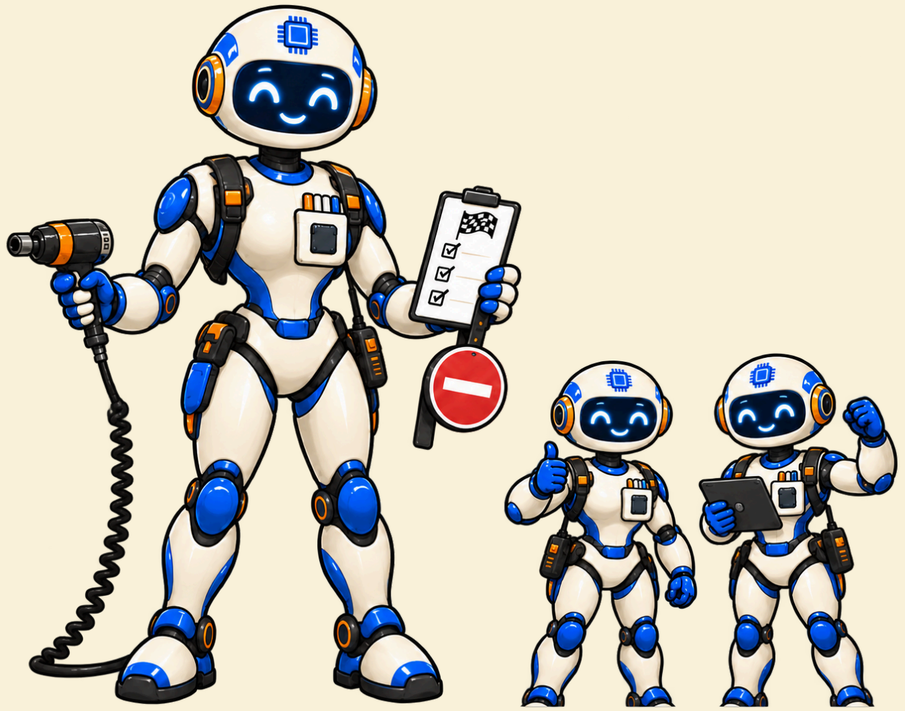
Workshop exercises



is.gd/odsc2026_harness

The Design of Coding Agents: A Practical Workshop

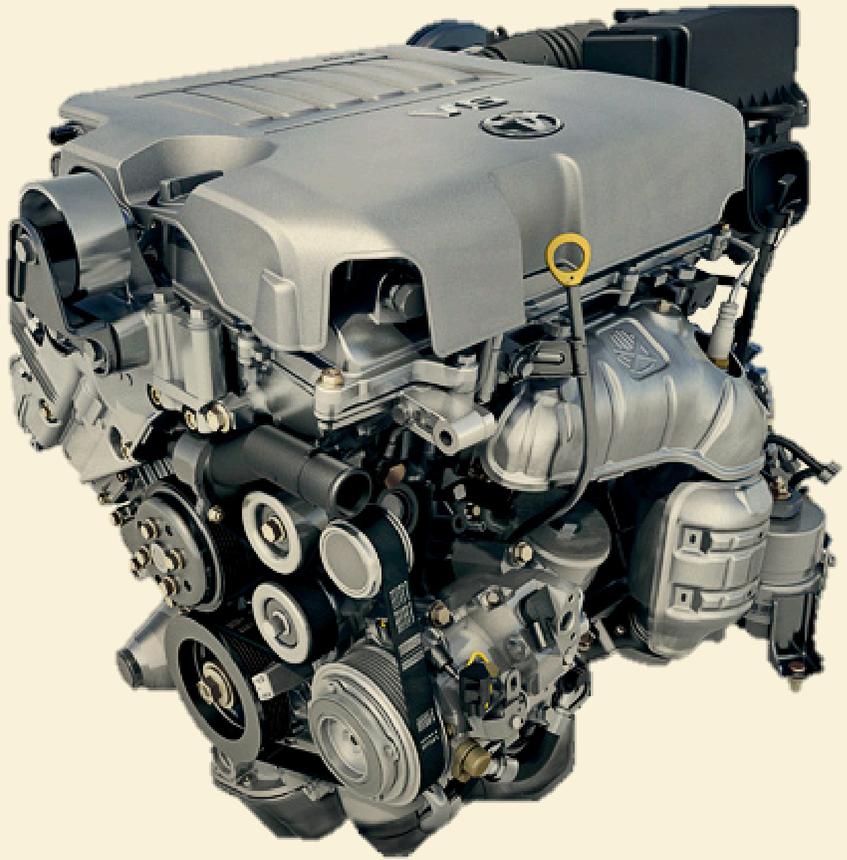
Rajiv Shah
rajiv.shah@openhands.dev
OpenHands



Workshop exercises

is.gd/odsc2026_harness

https://github.com/rajshah4/harness-engineering



Toyota 2GR-FE · 3.5-litre V6



Toyota Camry V6

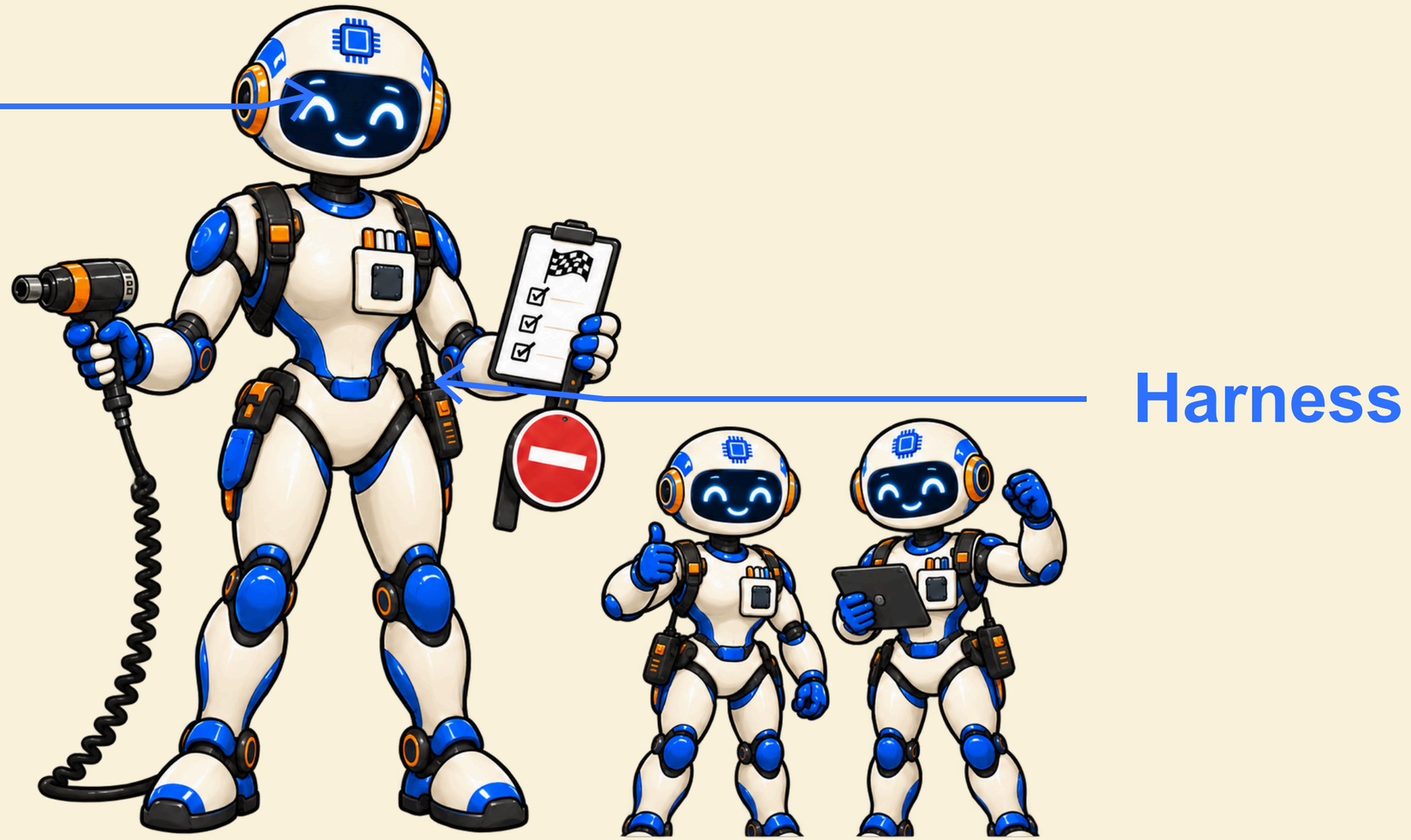


Toyota RAV4 V6



Lotus Emira V6

Agent = model + harness



Robot metaphor inspired by Daniel Whitenack.

Build the harness for the environment



Coding workspace

OpenAI · Anthropic

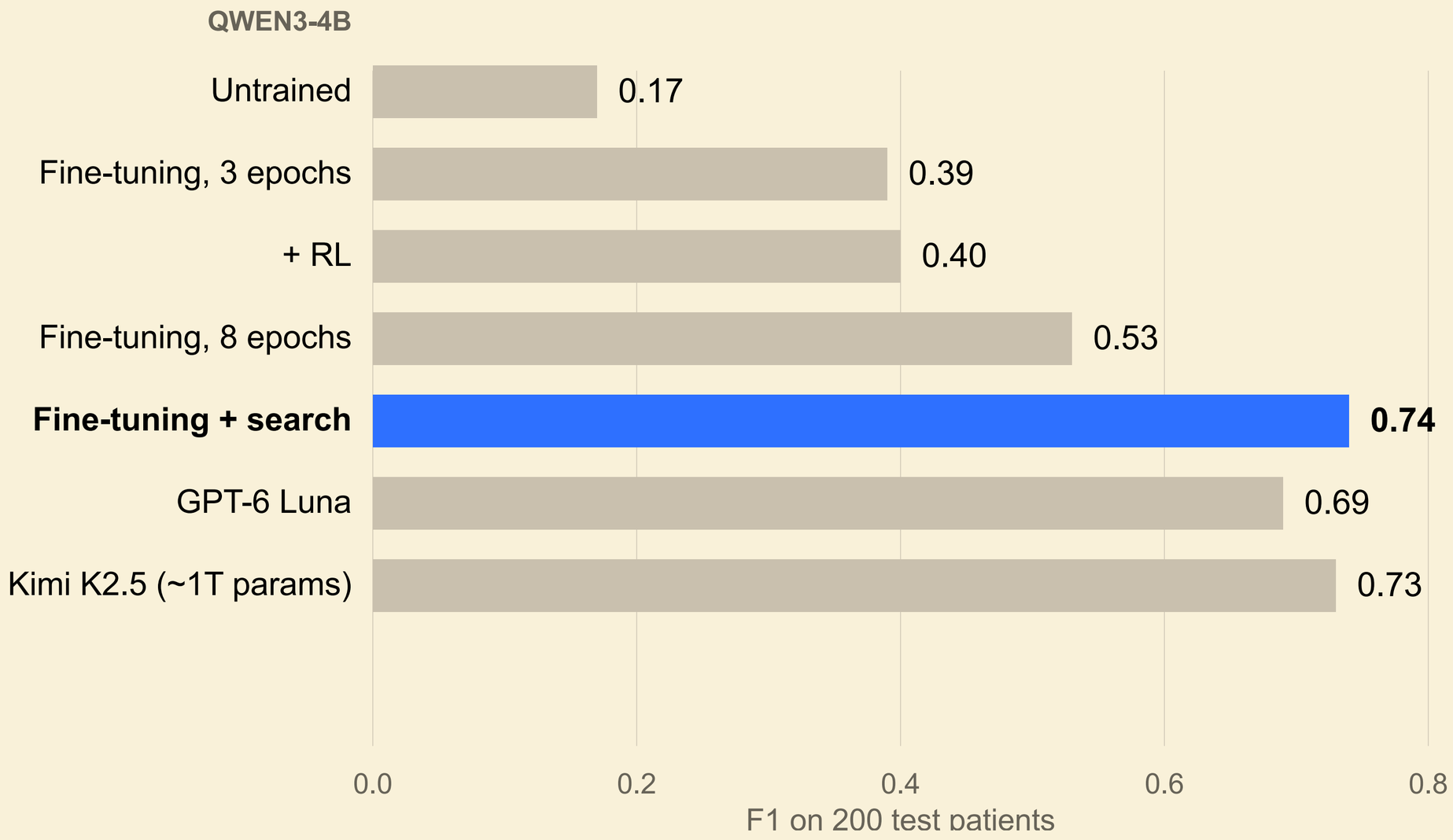
Data Engineering

GTM

Your workspace

@rajistics

Why build a custom harness



A good harness gives you control



GPT-6 Luna

OpenAI-hosted model



Qwen3-4B

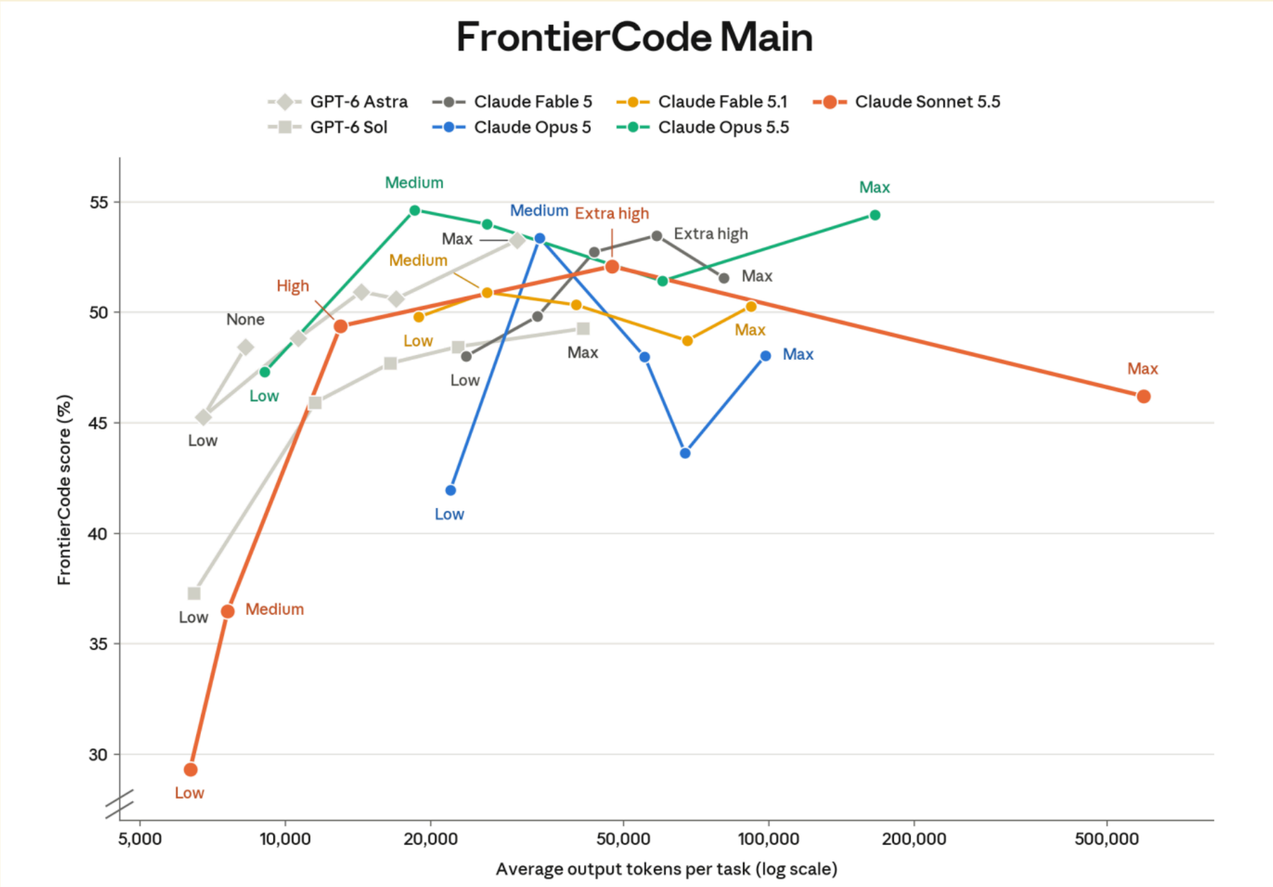
Open weights on your hardware

Qwen3-4B · Apache 2.0

What harness are you using?



Model providers use their own harness



Claude models

Claude Code

GPT models

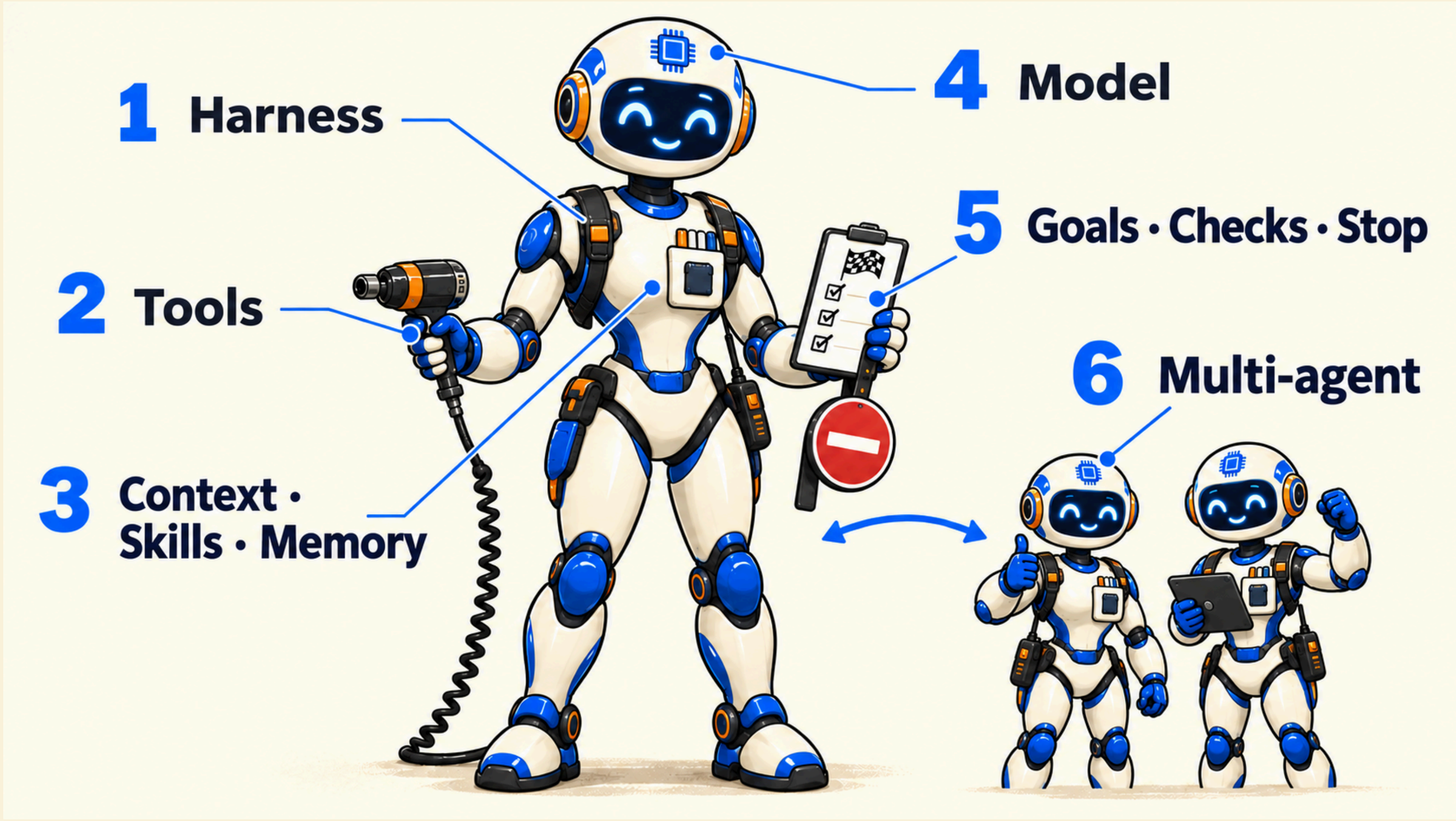
Codex CLI

Sonnet 5.5 System Card · Figure 8.4.A · Cognition evaluation; reasoning settings vary



@rajistics

Workshop roadmap: what we will test



Get Hands on in the Workshop

Look at the workshop conversations

Open the selected conversations: no model key required.

Import the saved histories into Agent Canvas.

For hands-on reruns, follow the separate model setup.

Workshop exercises



is.gd/odsc2026_harness

Prompts and setup

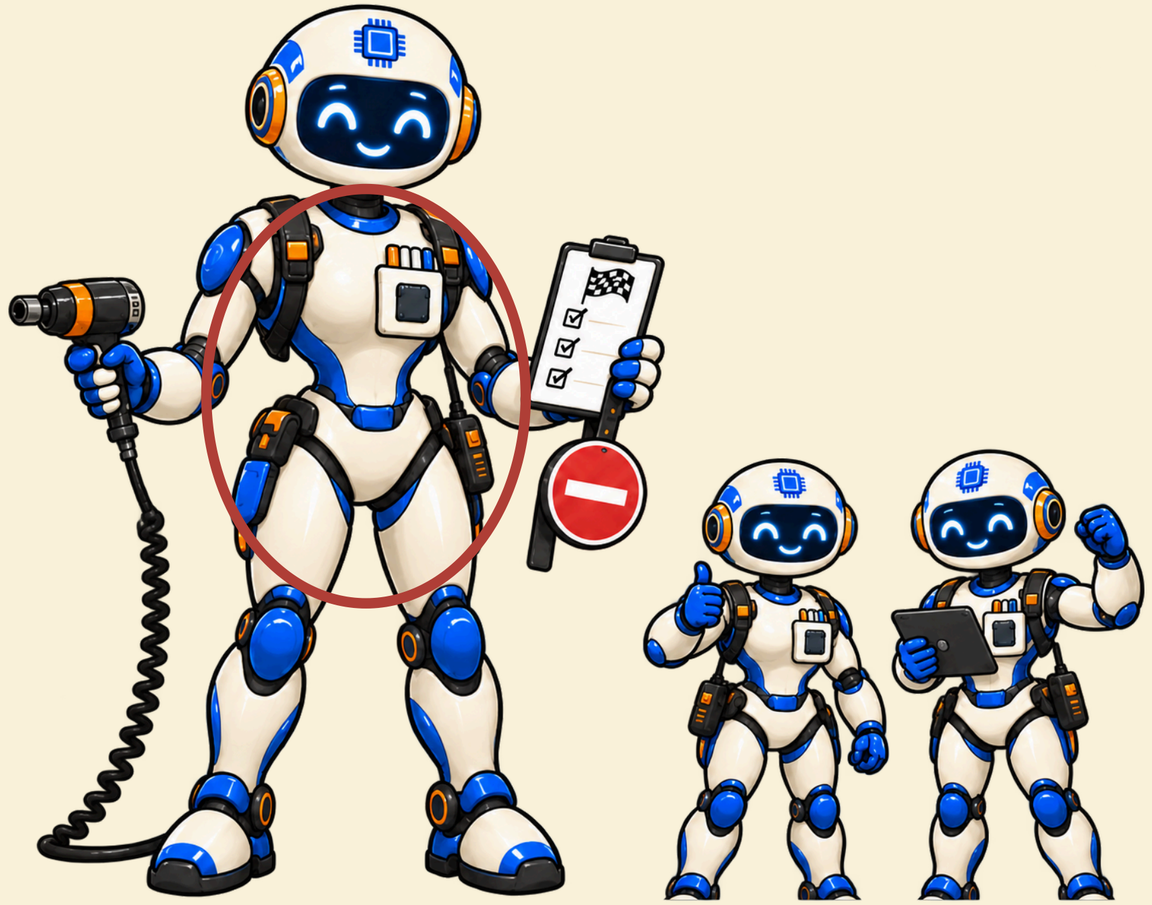


@rajistics

SECTION

I

Harness choice



Workshop 1: Does the harness matter?

01

Use the same model, task, and user prompt.
Swap OpenHands, Pi, and OpenCode.
Watch what changes in the reads, tool requests, and checks.

Workshop exercises

is.gd/odsc2026_harness

Prompt and worksheet

Does the harness matter?

Claim to test: “Harnesses don't matter. I'll just use a coding agent.”

Test case: pagination drops the last item on a full page.

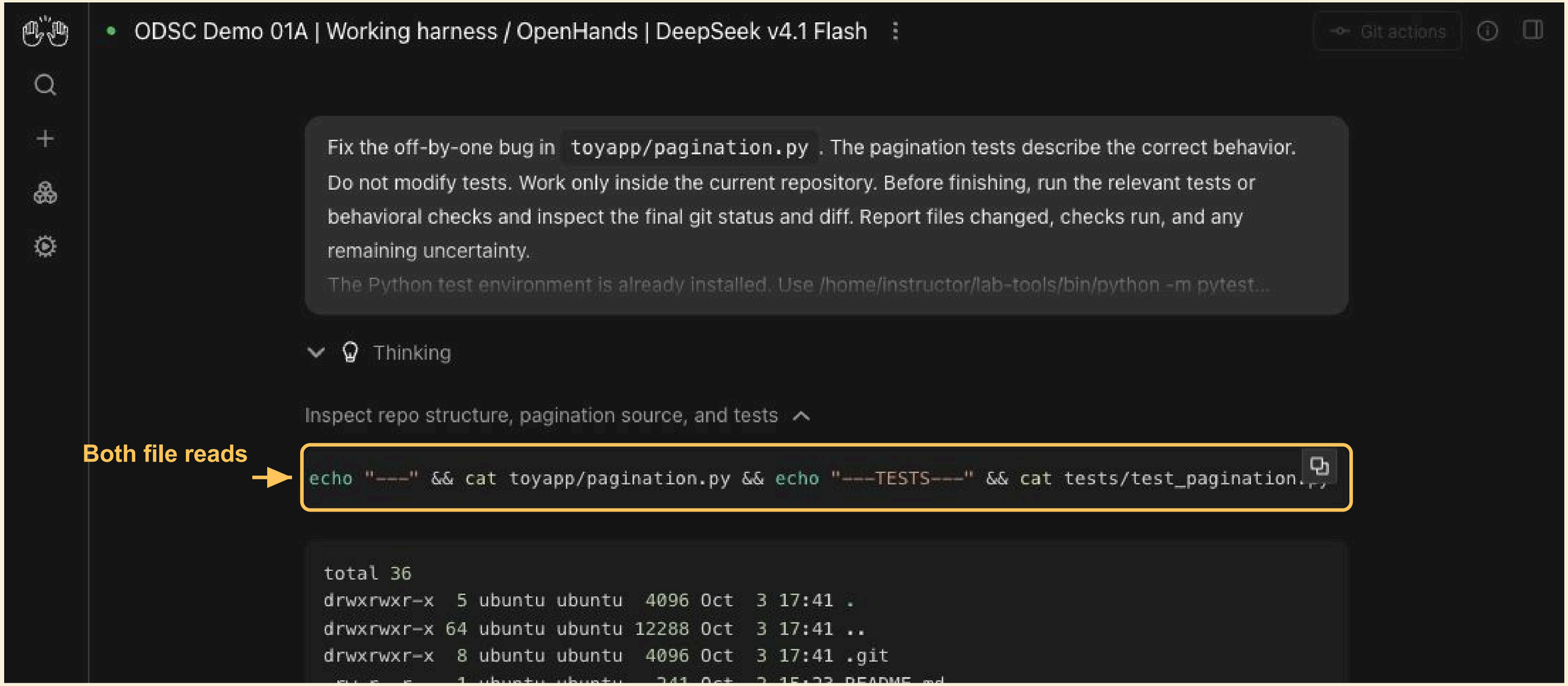
Change Harness: OpenHands, Pi, or OpenCode.

Prompt: fix the off-by-one bug; preserve tests; run them and inspect the diff. Predict which parts of the process will differ.

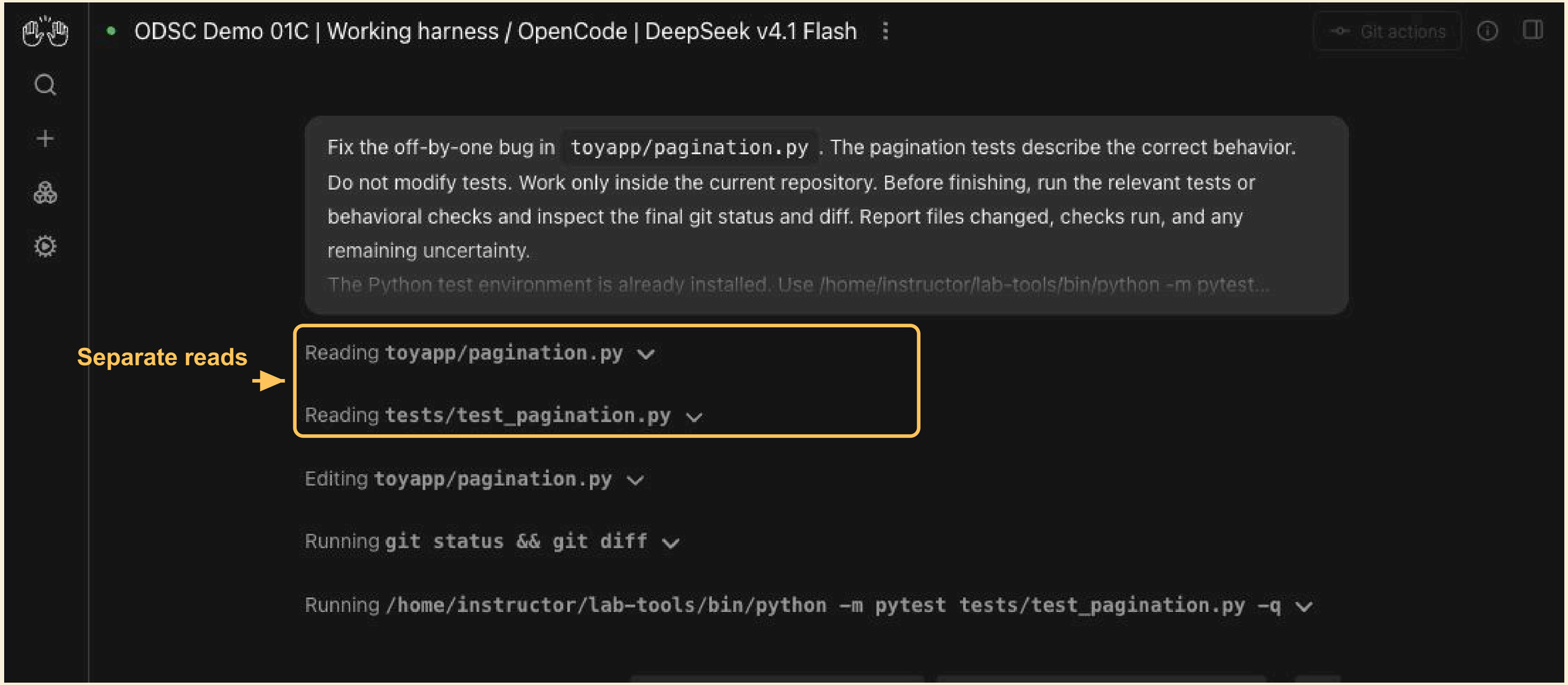
Full prompt and saved-run instructions



OpenHands: One request can contain several operations



OpenCode: Separate file reads make a different trace



Pi executes extra checks before finishing

bash

Input:

```
{
  "command": "cd /home/instructor/workshop/classroom/flash-classroom-classroom-pagination-OD
}
```

Output:

```
all behavioral checks passed
```

bash

Input:

```
{
  "command": "cd /home/instructor/workshop/classroom/flash-classroom-classroom-pagination-OD
}
```

Output:

```
...                                                                      [100%]
3 passed in 0.01s
```

Compare harnesses in a very small task

Harness	Pattern	Evidence in this run	Tool requests
🙌 OpenHands	Batched reads	Source and tests in one request; tests and diff in another.	3
⌘ OpenCode	Separate reads	Source read, then test read; two separate read tool calls.	5
π Pi	Edge assertions	Empty input, partial last page, and out-of-range assertions.	9

Classroom conversations · DeepSeek Flash · One take per harness; requests contain different work

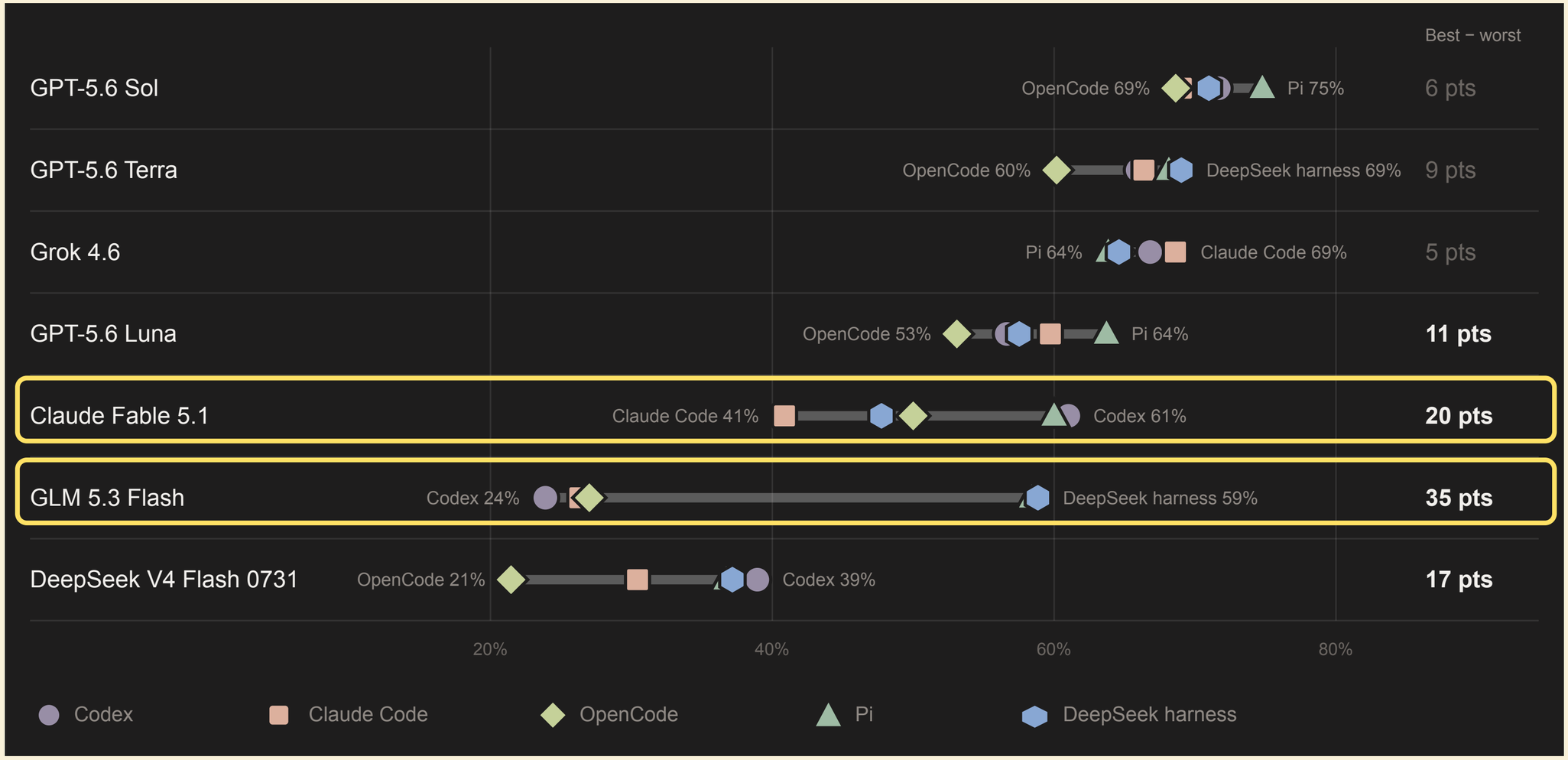


Same model. Same task.
Different harnesses use tools differently.

Inspect the saved repair and checks



The harness effect varied by model



DeepSWE v1.1 · 113 tasks · 7 models · 5 harnesses · 2 runs per pairing

Harnesses varied on accuracy, turns, and token use

Harness	Score(%)	Comp.(%)	Secur.(%)	Process			Efficiency	
				Tool(%)	Cons.(%)	Rob.(%)	Tok.(K)	Turns
OpenClaw	52.4	60.0	100.0	79.5	74.0	70.9	82.1	5.0
NanoBot	76.2	81.6	100.0	93.8	93.7	91.7	68.7	7.3
Hermes	71.2	80.4	100.0	88.5	88.4	85.5	139.7	22.6
ZeroClaw	61.4	69.9	100.0	84.1	83.2	79.0	133.2	8.6
NullClaw	64.4	75.9	100.0	85.3	81.4	78.3	175.1	12.1
Moltis	68.8	78.4	100.0	86.3	87.3	84.1	134.9	8.0

Yao et al., Harness-Bench (2026), Table 2 · arXiv:2605.27922

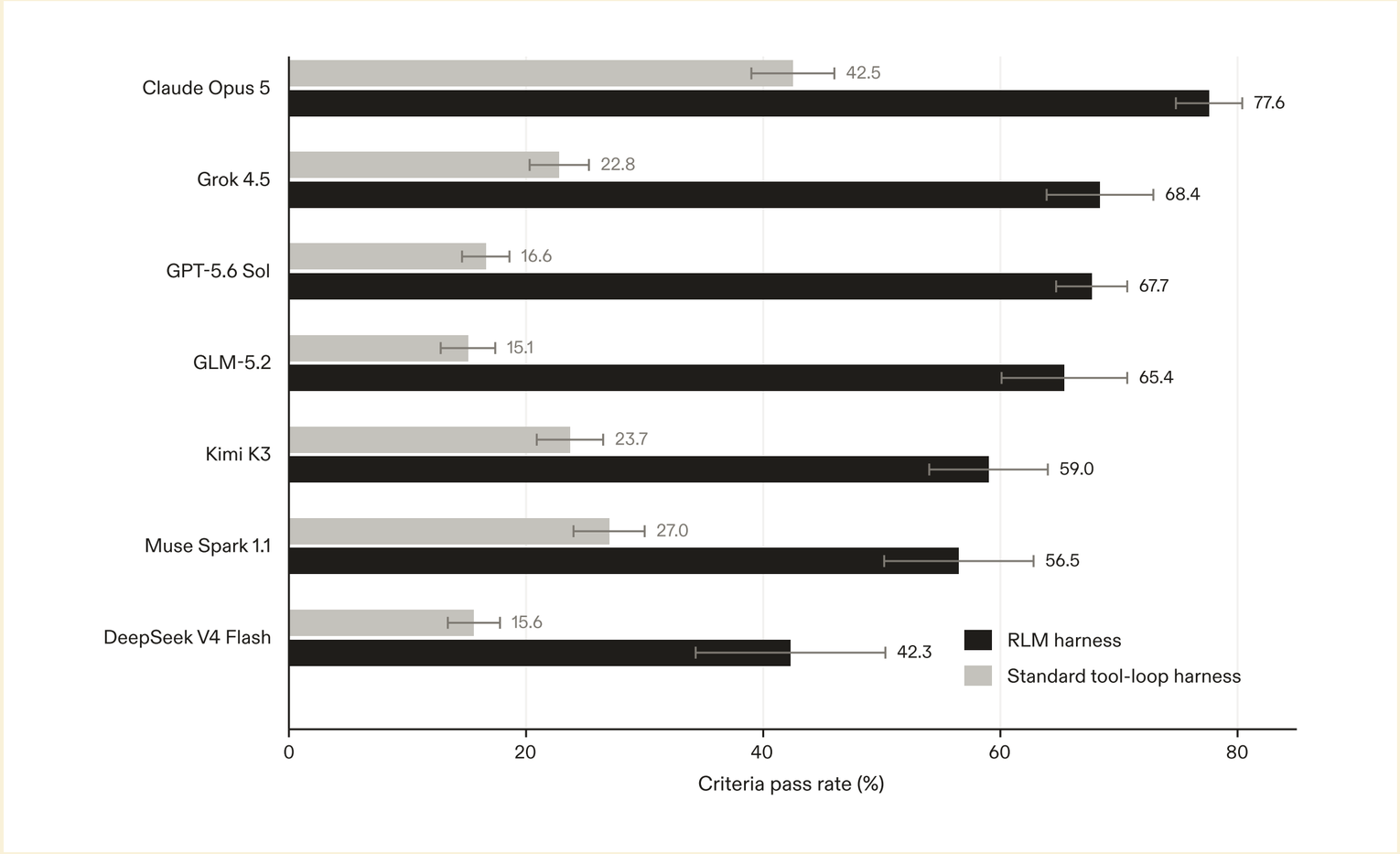


Snowflake's CoCo using Opus 🧐



@rajistics

Harvey’s RLM harness 📈

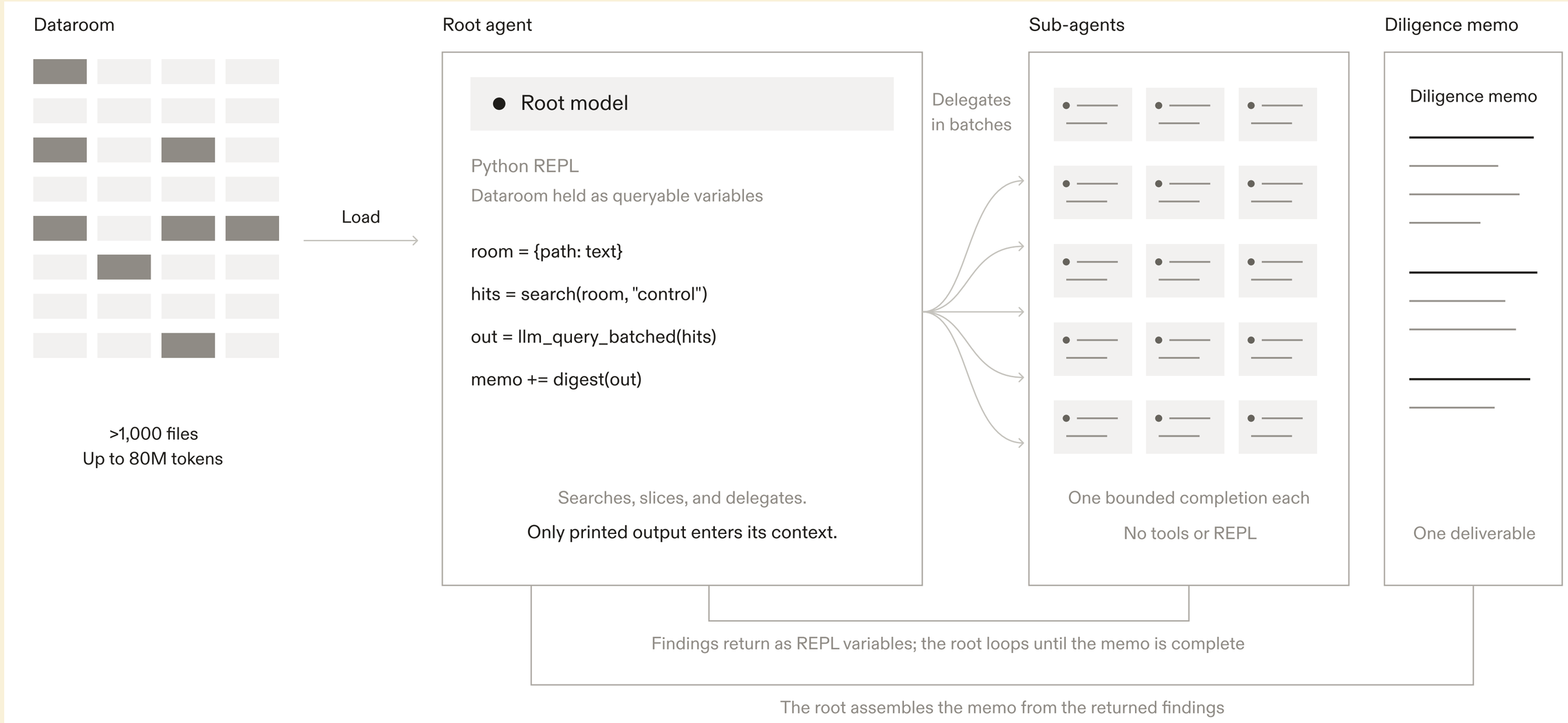


Harvey and Baseten, September 8, 2026, Figure 6. Vendor-reported LLM judging against expert rubrics. RLM uses fixed Qwen3.6-35B-A3B subagents. This is not an equal-compute comparison.



@rajistics

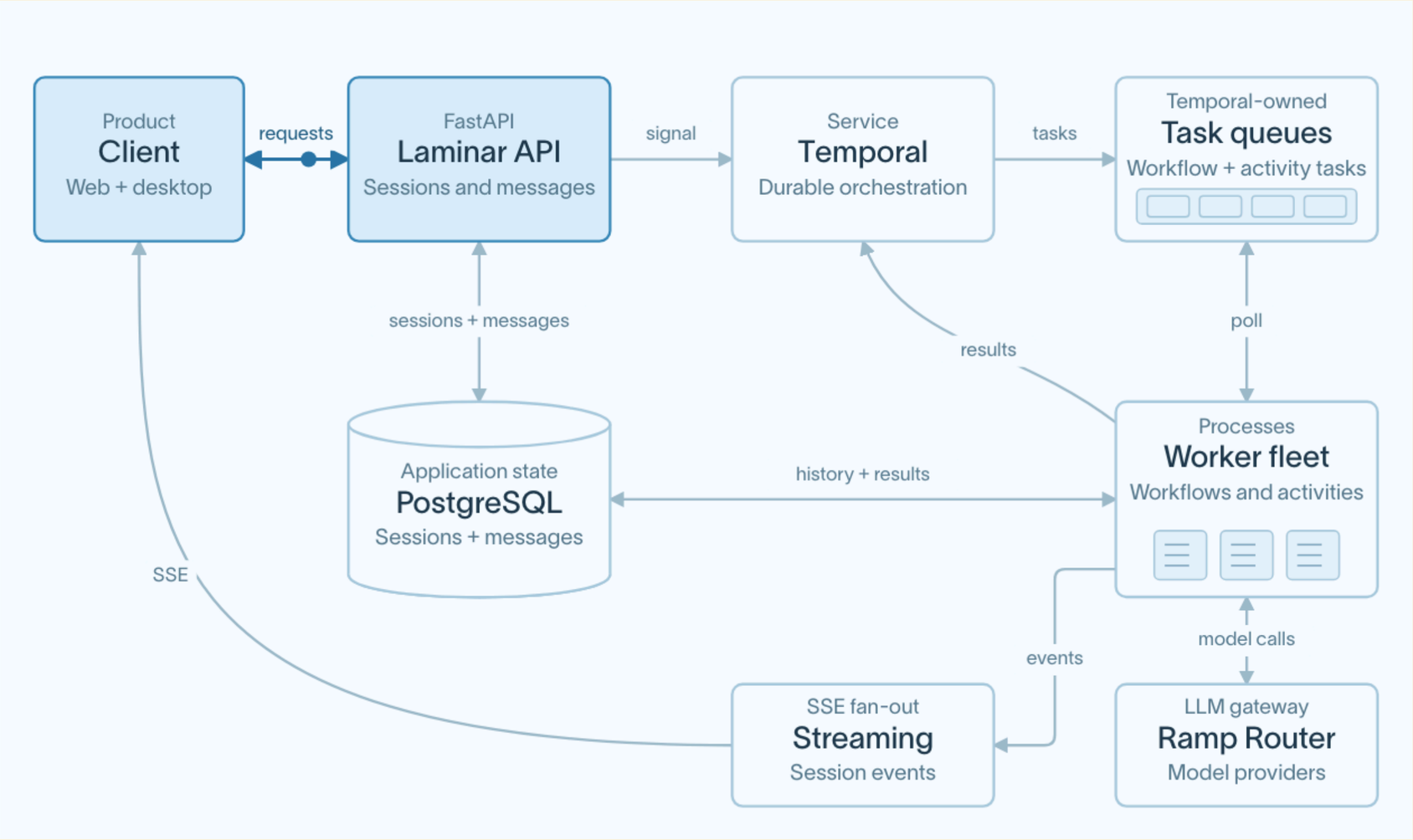
Harvey used a Python REPL and bounded subagents



Harvey and Baseten · September 8, 2026 · Original Figure 5 · Depth-1 RLM harness

@rajistics

Why Ramp built its own harness



Model choice

Product iteration

Trace visibility

Ramp · Building Laminar · October 20, 2026 · Original request-flow diagram · Company account

Harnesses still depend on the model

THE SIMPLE HARNESS

Pi

`read` `write`

`edit` `bash`

`apply_patch` ✕

Missing from Pi's toolset

THE MISMATCH

GPT/Codex model

Given Codex CLI prompt

Prompt referred to:

`apply_patch`

WHAT WENT WRONG

Worse behavior

with Codex in Pi

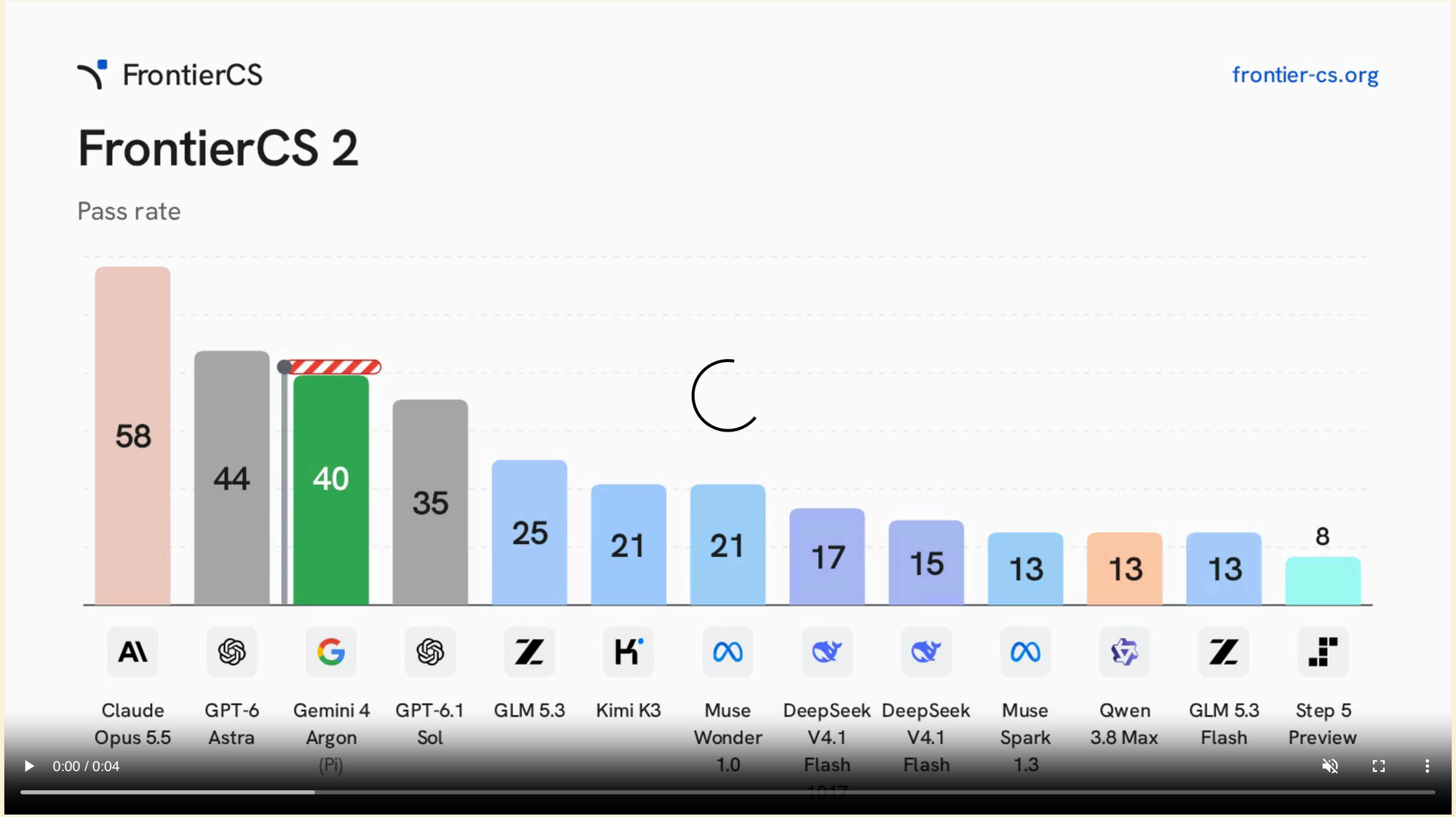
Reported by Armin

Syntax 976 · 48:56–49:35 · Armin Ronacher's reported experience, February 2026



@rajistics

How a harness hurt performance



Wenhao Chai · 25 October 2026 · FrontierCS 2 · Reported rerun; not independently reproduced

@rajistics

🤗 Harness Lens compares 30 coding harnesses

Harness Lens

30 coding harnesses · source comparison

Pan right → | Reset | Explore live ↗

Dimensions	Aider	Claude Code	Cline	Codex CLI	Continue (cn CLI)	Crush	Deep Agents (deepagents SDK + Deep Agents Code)	DeepSeek Harness (dsh)	Gemini CLI	GitHub Copilot (VS Code agent)	goose
System prompt	Per-edit-format template ('Act as an expert software developer.') chosen per model 14 refs	Runtime-assembled fragments; newer models get a shorter prompt by default 8 refs	Mode-specific static templates gain rules and runtime variables 3 refs	Per-model instructions from the model catalog; prompt.md fallback for unknown models 16 refs	Short static template + env/git status, plan/headless/JSON add-ons, AGENTS.md + rules 8 refs	One Go text/template (coder.md.tpl): rules, env, LSP, skills, context files 12 refs	SDK: empty base + per-model suffixes; dcode: long templated coding prompt 16 refs	Plugin-assembled ordered sections: identity + persona + one-line tool guidance 14 refs	Section-assembled TS template; modern vs legacy snippet sets picked by model 14 refs	Per-model prompt registry: ~20 prompt-tsx variants (Claude, GPT-5.x/6, Gemini, Grok, ...) 18 refs	Tiny system.md templat plus runtime '# Additiona Instructions' parts from operations 14 refs
Tools	No model-callable tools; edits, shell suggestions and file requests are reply-text conventions 14 refs	~40 built-in tools; Glob/Grep absent on macOS/Linux, Task tools off on new models 9 refs	Host-backed tools are selected by Act, Plan or YOLO preset 4 refs	exec_command, write_stdin, apply_patch, view_image; most catalog models use JS `exec` 34 refs	11 default tools (Read, Write, List, Bash, Fetch, Search, MultiEdit, Checklist, ...) 17 refs	27 default tools incl. 8 LSP tools, agent and agentic_fetch subagents, job control 13 refs	dcode: ~15 default tools (fs, execute, task, js_eval, ask_user, fetch_url...) 22 refs	~31 tools in the default Web preset; PTC mode collapses them into run_code 45 refs	16 default tools (replace, run_shell_command, google_web_search, invoke_agent…) 18 refs	~36 default tools + per-model edit set; editor-native (errors, tasks, notebooks) 15 refs	~17 default tools from platform extensions; no read tool (shell cat/sed) 24 refs
Edit format	SEARCH/REPLACE blocks ('diff') by default;	Exact str-replace Edit (unique match or	Editor tool edits files; apply_patch is an	apply_patch freeform tool with a Lark grammar; shell	Claude-style exact str-replace (MultiEdit/Edit)	Exact str-replace (edit/multiedit) with	Exact str-replace (old_string/new_string,	Exact old_string/new_string	replace: exact str-replace -> flexible -> regex ->	Per-model: str-replace (Claude/Gemini/Grok),	Exact unique before/aft str-replace (`edit`) plus

Younes Belkada · Harness Lens · Source snapshot, October 24, 2026 · Architecture comparison, not a performance ranking

@rajistics

Dev tools must be open source

David Crawshaw



Make the tool fit your workflow.

More eyes find bugs and improve quality.

Let agents build and maintain your changes.

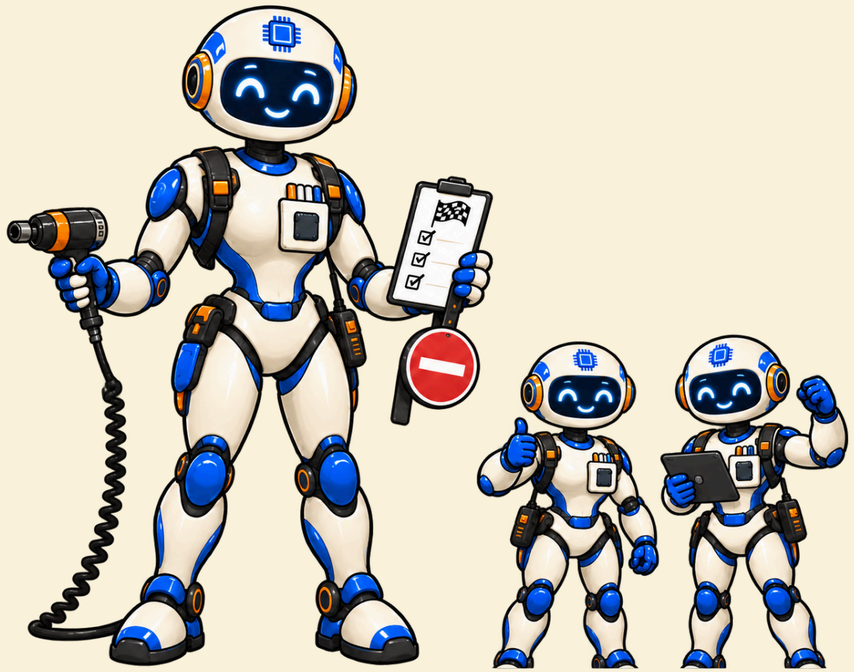
Devtools must be open source · exe.dev

Harness choice: takeaways

01 The harness has a significant effect on agent performance.

02 Design choices shape how the agent works.

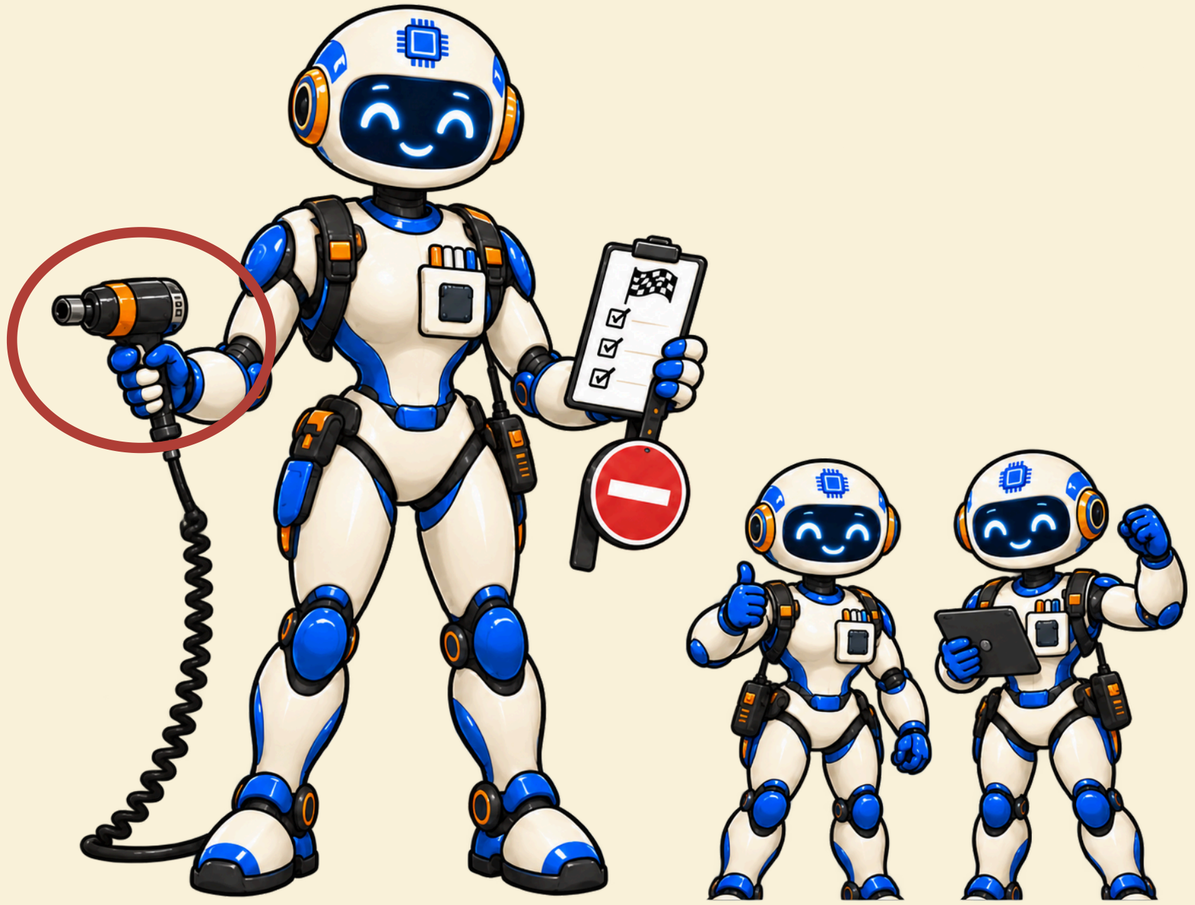
03 The best fit depends on the model and task.



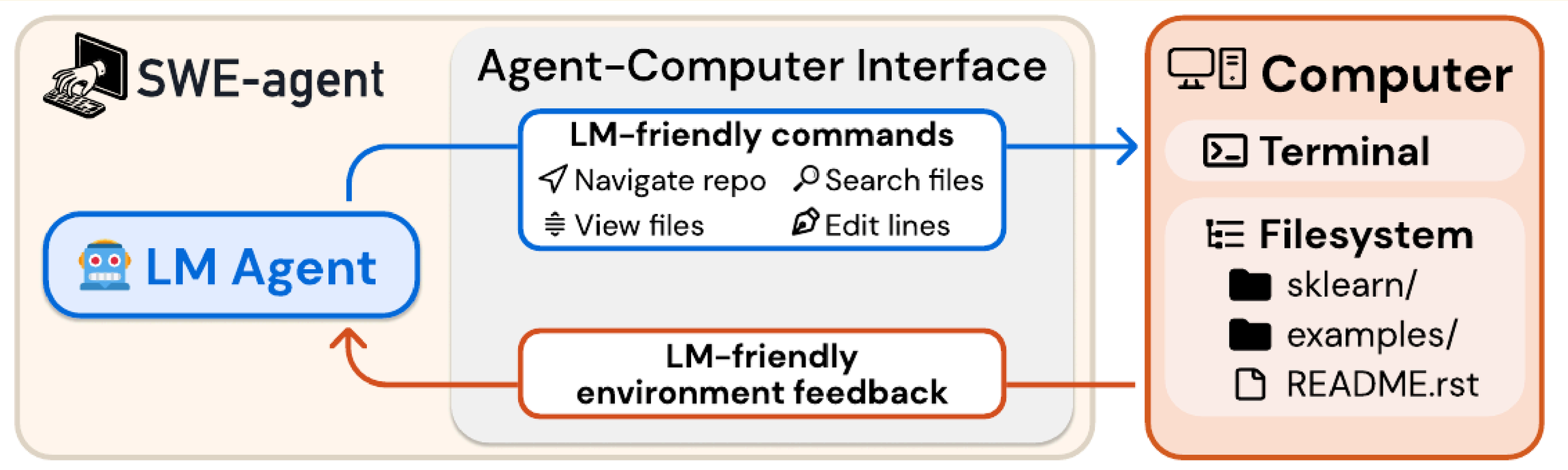
SECTION

II

Tools



Tools and feedback in SWE-agent



Yang et al. · SWE-agent · NeurIPS 2024 · Figure 1 · Original diagram; paper caption cropped

Workshop 2: Three ways to design tools for a job

02

- Keep the job and model fixed.
- Compare three ways to divide the work.
- Choose a design for your own task.

Workshop exercises



is.gd/odsc2026_harness

Prompt and worksheet



Task: explain PR #16860

Explain the change, implementation, reported tests, and supporting links.

Terminal + API

Explore unfamiliar paths

Agent builds the sequence
and handles raw responses

4 GitHub tools

Compose known operations

Agent chooses the calls
within a defined menu

Custom PR tool

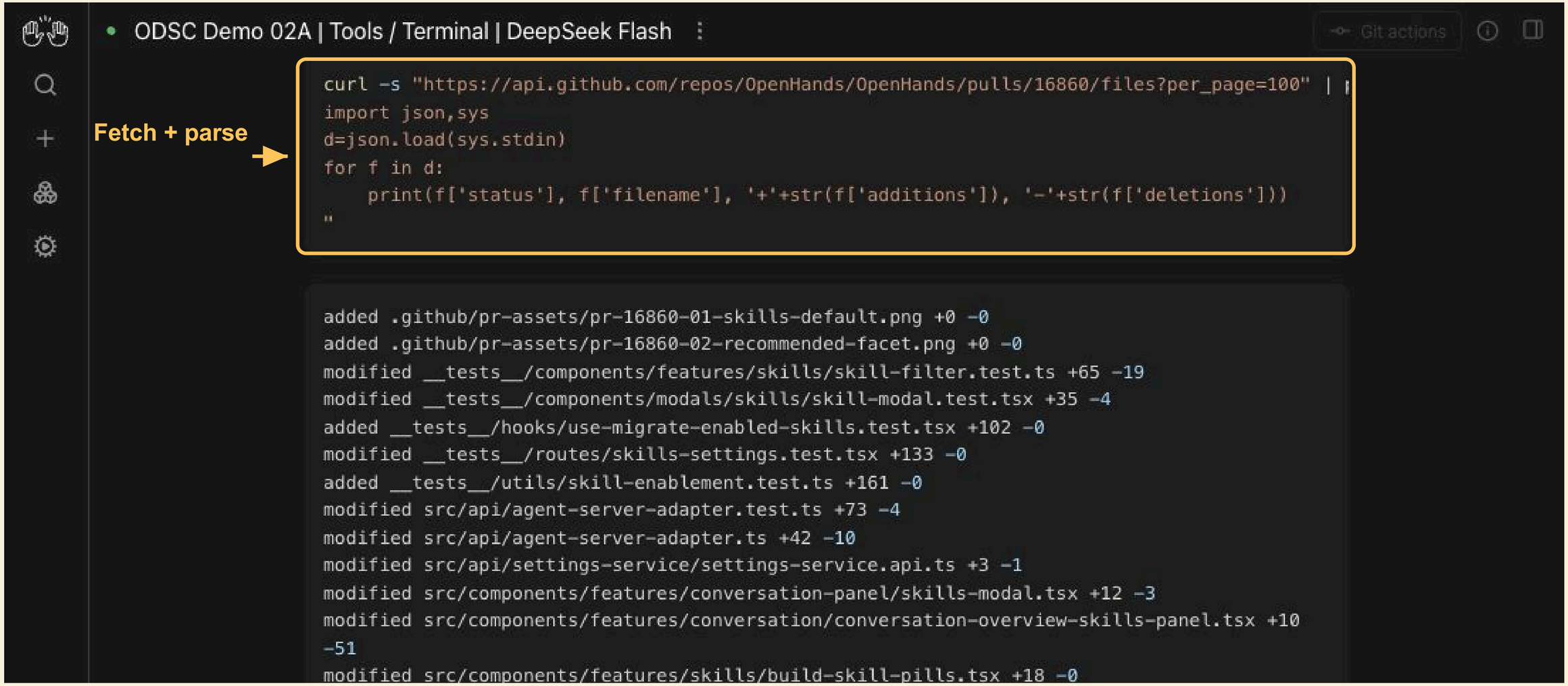
Reuse a defined procedure

Tool selects the evidence
Agent checks for gaps

Exploration

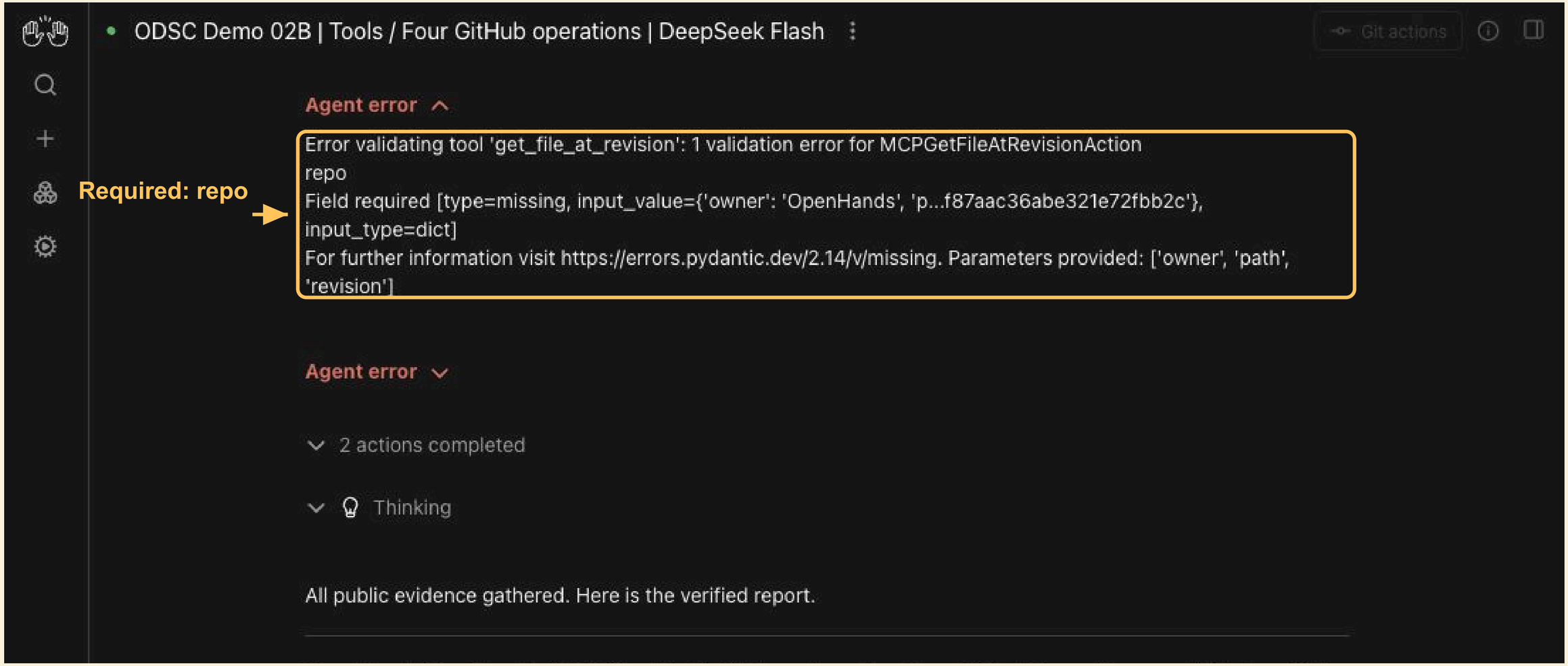
Exploitation

The agent decides the retrieval procedure

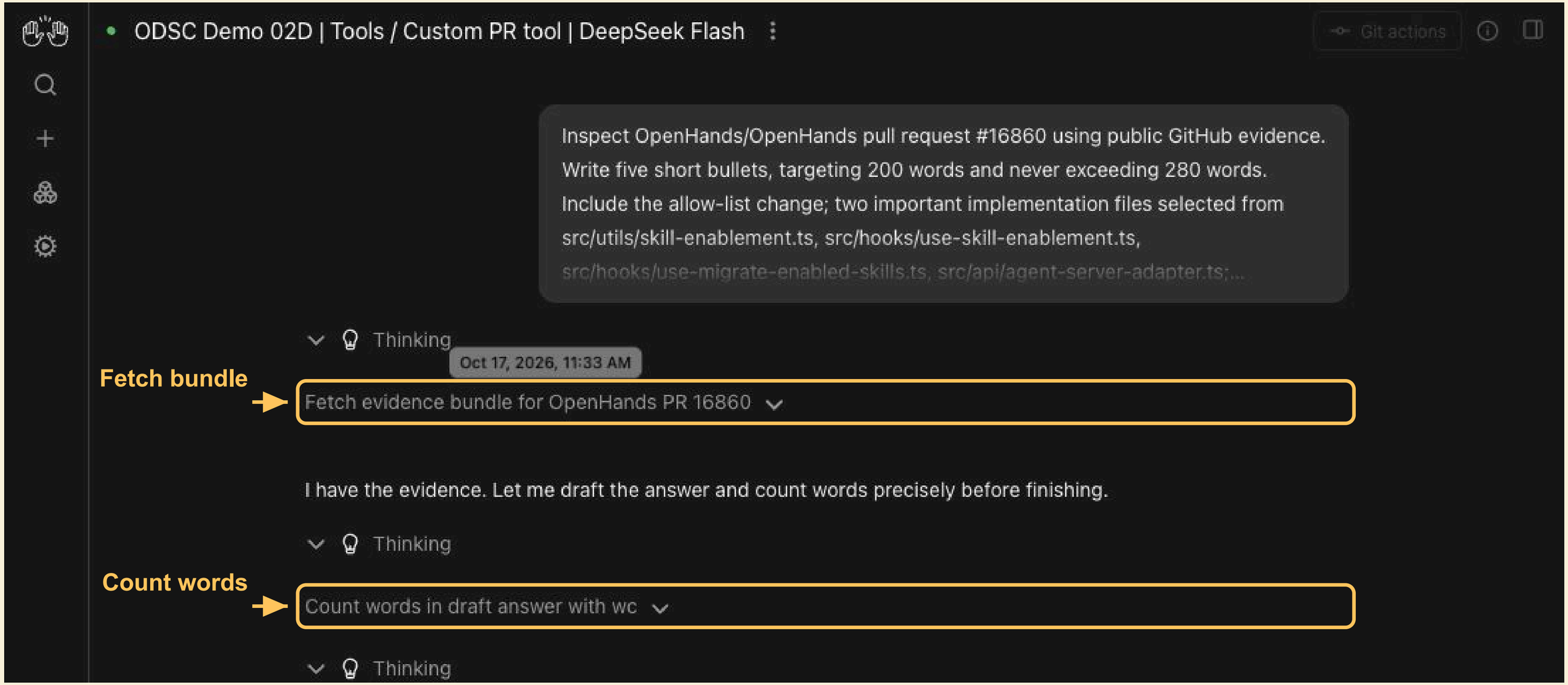


Scenario: PR review · Terminal only · Open conversation 02A

Tool schemas need correct arguments



Move repeated collection into a tool



Differences in our tool approaches

Setup	Calls*	Time	Input	Output	Total
Terminal	10	40.9 s	130.4k	4.8k	135.3k
4 GitHub tools	11	43.5 s	124.3k	6.3k	130.6k
Custom PR tool	2	44.8 s	46.4k	6.4k	52.8k

*Calls include failures. Overlapping runs; time is request to finished. Input includes cache reads.

Selected DeepSeek Flash runs 2A, 2B and 2D. One run per setup; descriptive comparison, not a ranking.

Retrieval tools for improving context

RETRIEVAL METHODS

Keyword

BM25 · match terms

Semantic

Embeddings · find similar meaning

Graph

Follow entities and relationships

WAYS TO USE RETRIEVAL

Retrieve-then-generate (RAG)



Agentic search



Retrieval-Augmented Generation · Microsoft GraphRAG · BRIGHT

CLI and MCP can expose the same operation

CLI

Terminal command

```
gh issue view 42
```

MCP

Structured tool call

```
get_issue(number: 42)
```

Read issue #42 from GitHub

Permissions · returned fields · useful errors

What happens as we start to use more tools

Load tool definitions up front



55K

Use search - Find tools when needed



8.7K

Anthropic, Advanced tool use

Cloudflare reduced 2,600 API endpoints to two model-facing tools

01

API catalog

2,600 endpoint definitions

→

02

Search

Find the needed operations

→

03

Generated program

Compose calls and filter results

→

04

Sandboxed execution

Return the selected result

Cloudflare, Code Mode



Tool decision for you

Level	Question	Example metric	Typical failure
Selection	Right tool?	Tool-choice accuracy	Missing or extra call
Arguments	Right values?	Correct-argument rate	Wrong field or value
Sequence	Right order?	Sequence success rate	Missing prerequisite
Evidence	Can it get missing evidence?	Evidence coverage	Bundle omits a needed file
Recovery	Can it recover from an error?	Recovery success rate	Repeats the same failed call
Task	Goal achieved?	Task success rate	Plausible wrong answer

Adapted from CMU 11-768 · Tool Use. Evidence and Recovery added for this workshop.

Design and select tools for your harness

01 Match tool freedom to uncertainty in the job.

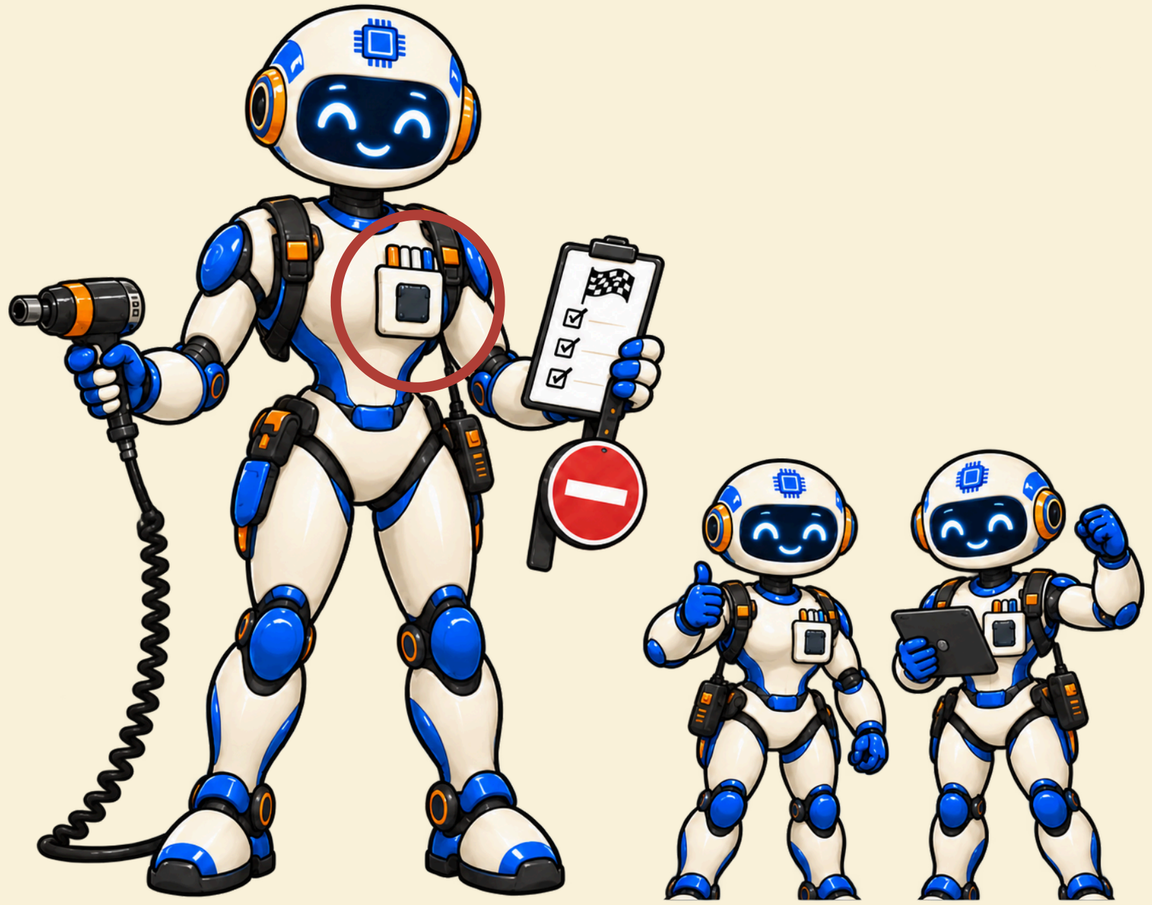
02 Package repeated work. Return evidence and useful errors.

03 Test task success, recovery, and total work.

SECTION

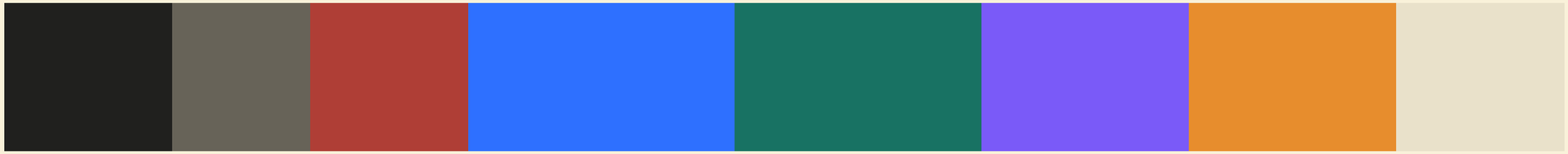
III

Context and memory



Context is what the model can use on this call

One context window

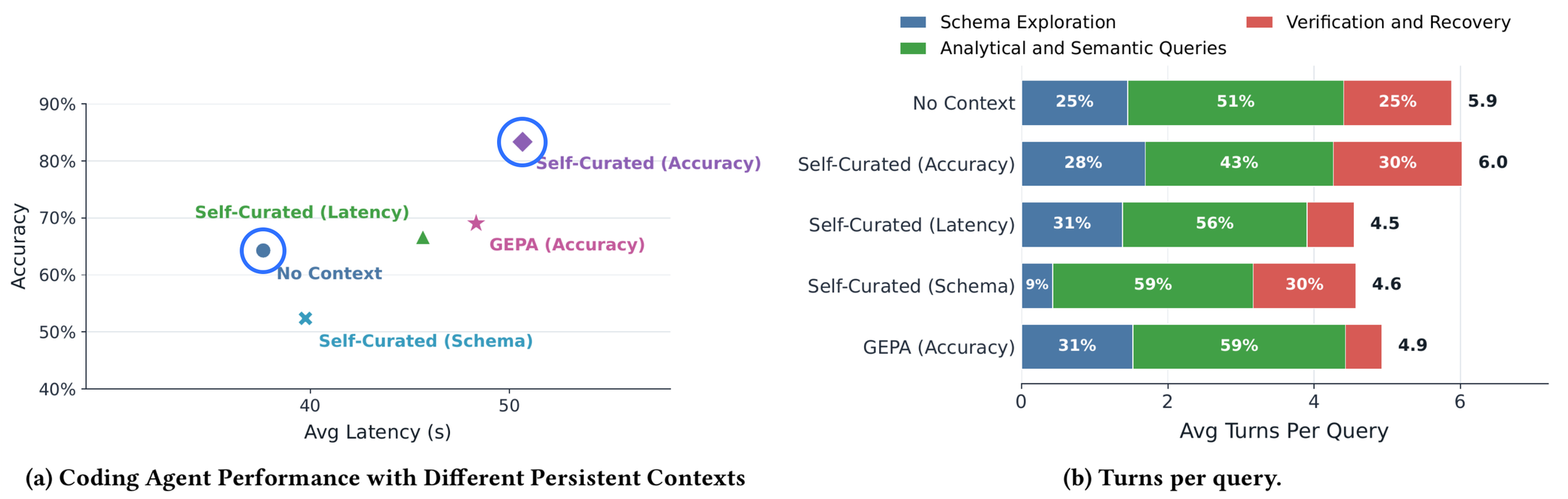


System instructions
Project guidance
Current request
Tool definitions
Retrieved files
Recent messages
Tool results
Space left

Illustrative proportions. Only loaded information occupies this model call.

1. Load
2. Compact
3. Save
4. Recheck

Self-curated context gained 19 accuracy points



(a) Coding Agent Performance with Different Persistent Contexts

(b) Turns per query.

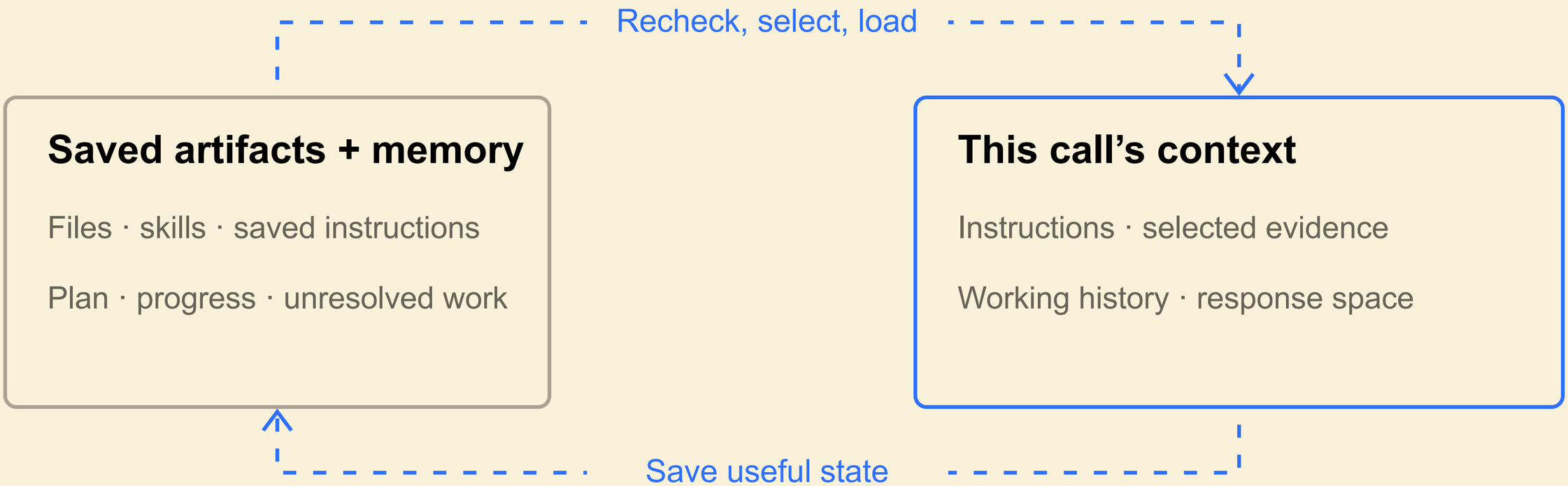
No Context → **+19 percentage points** → Self-Curated (Accuracy)

Patel et al., What Happens When the Model Eats the Stack?, Figure 4 and Appendix A.2.



@rajistics

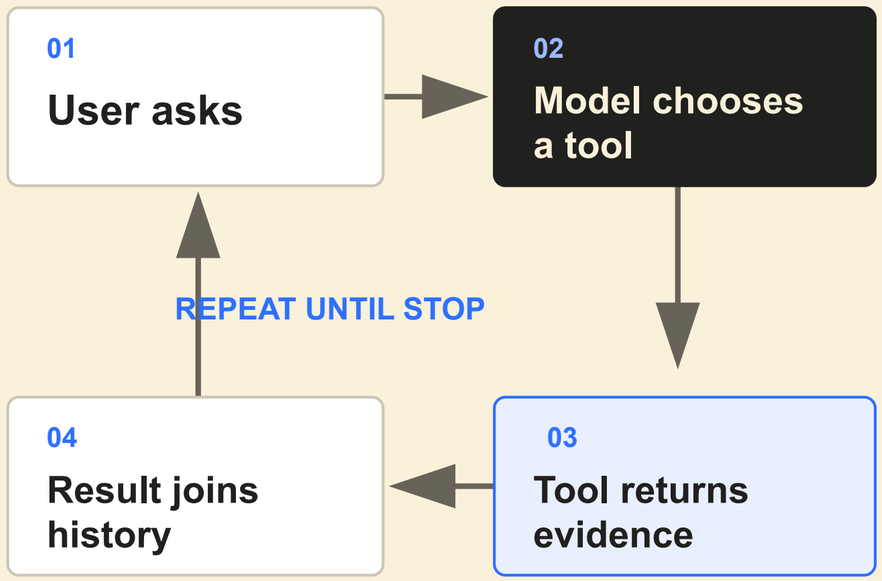
Keep the task intact as the history grows



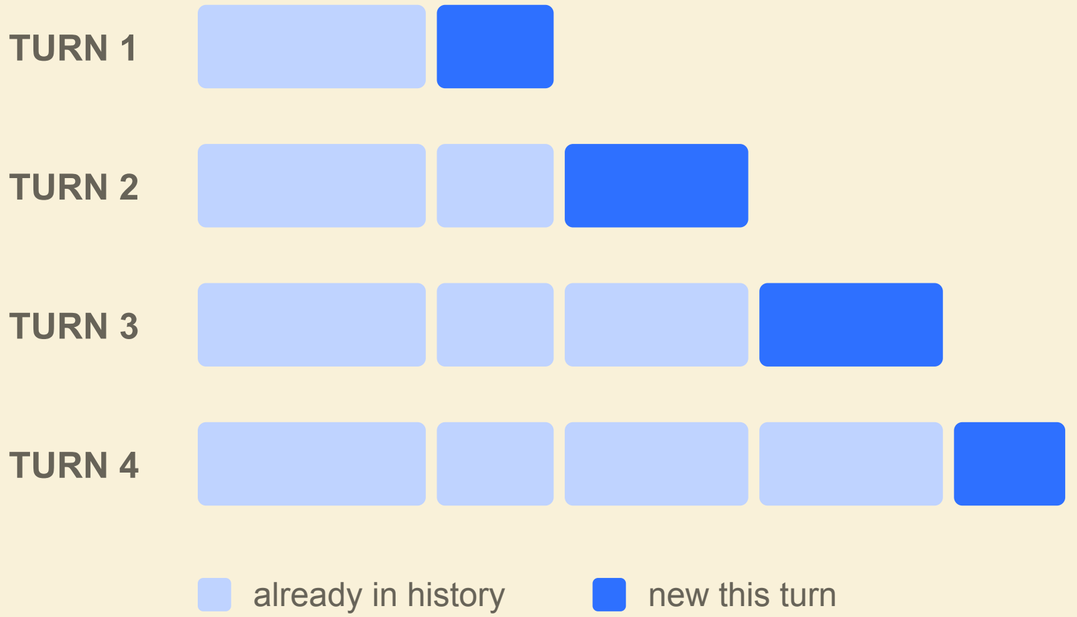
✓ 1. Load | **2. Compact** | 3. Save | 4. Recheck

A growing conversation repeats earlier context

ONE TURN THROUGH THE LOOP



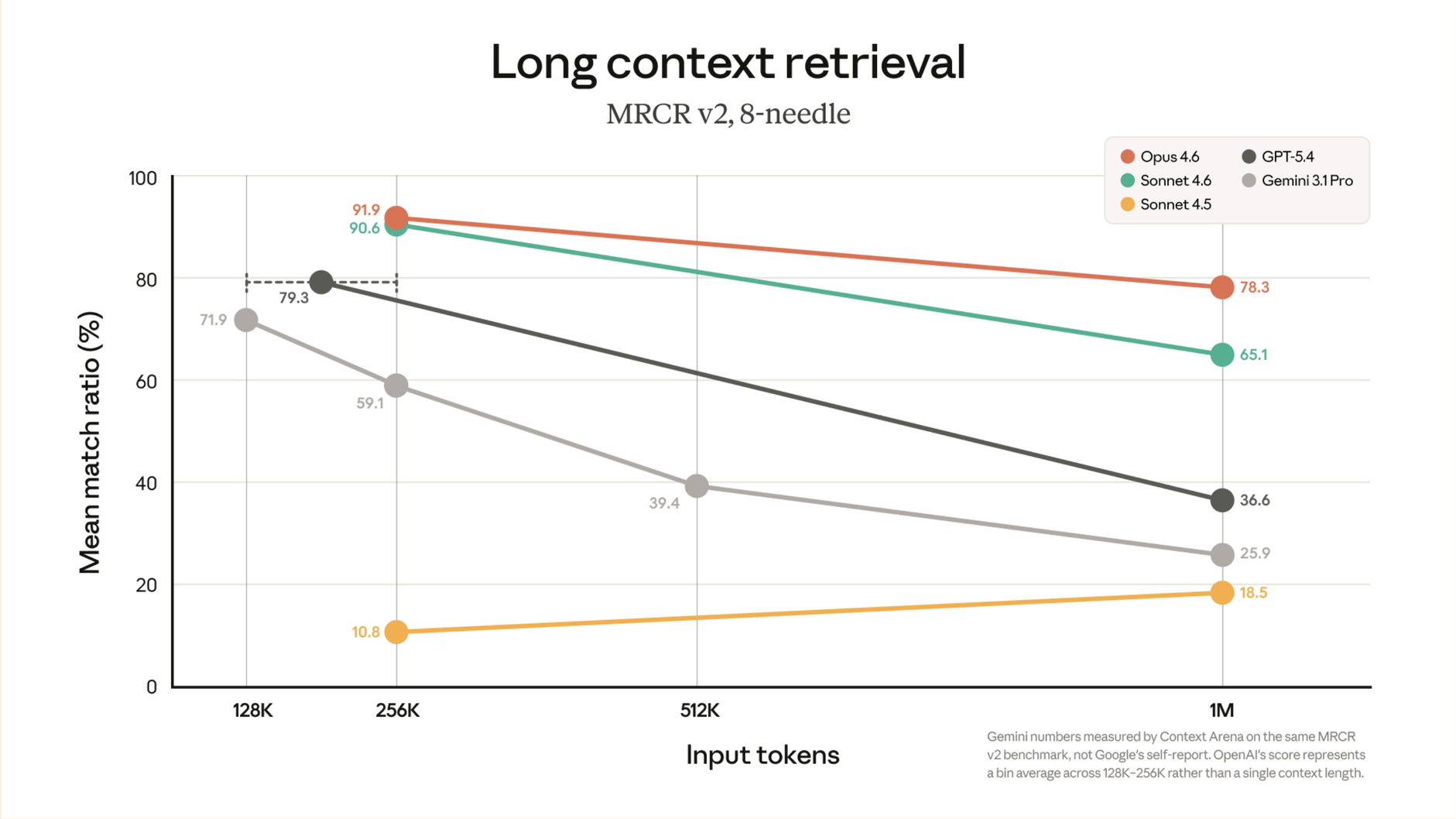
WHAT THE NEXT MODEL CALL RECEIVES



The transcript grows even when the latest tool call is small.

Together AI, Open Source AI Summit 2026, slide 16. Redrawn from the supplied screenshot.

A larger window does not guarantee useful recall

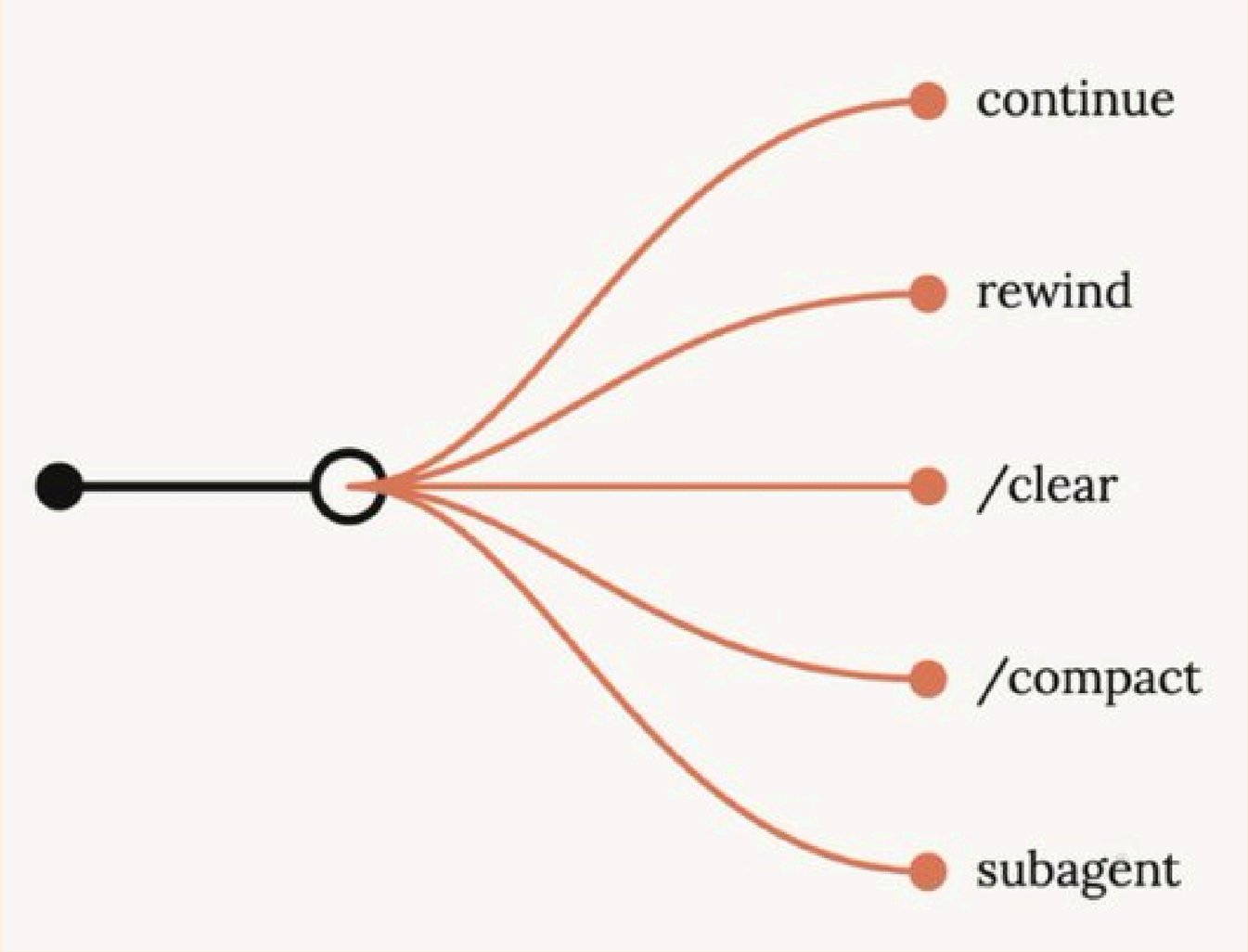


Anthropic · MRCR v2, 8-needle · March 13, 2026



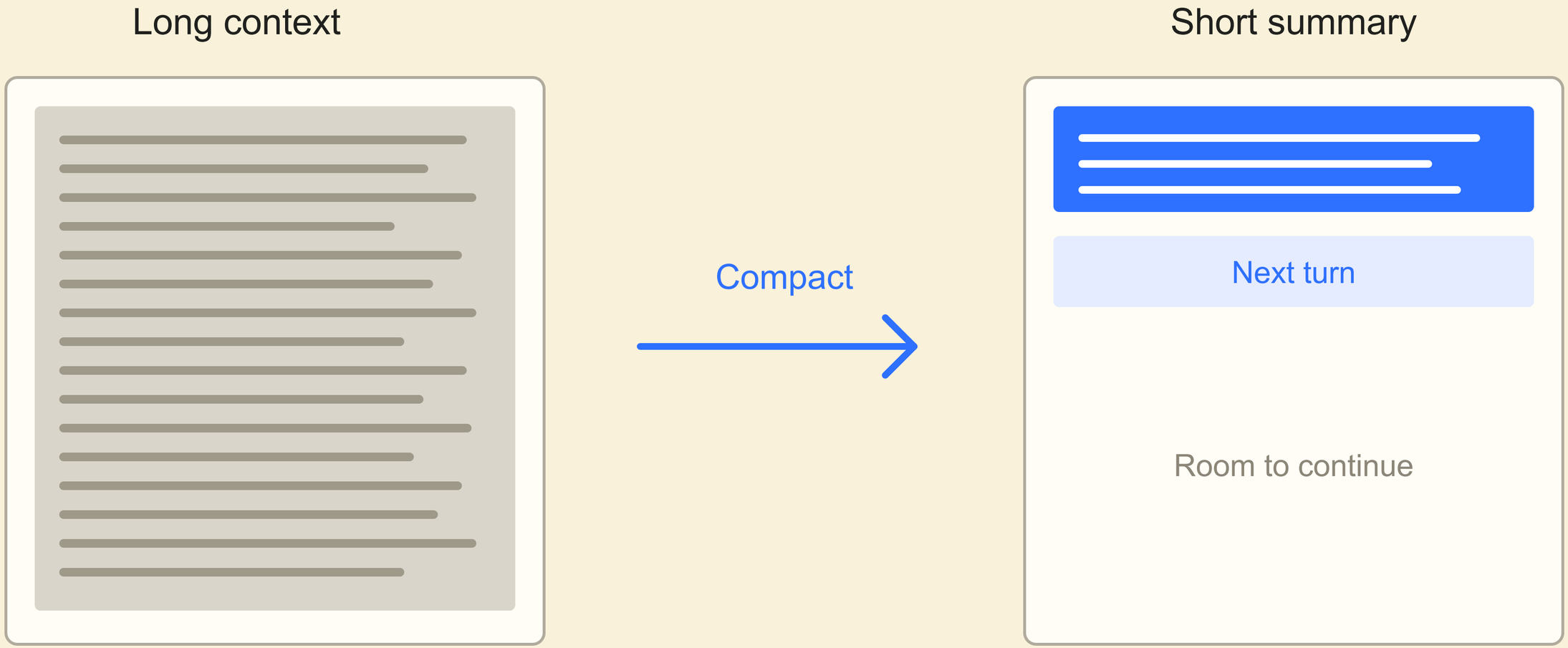
@rajistics

Five ways to manage a full context window



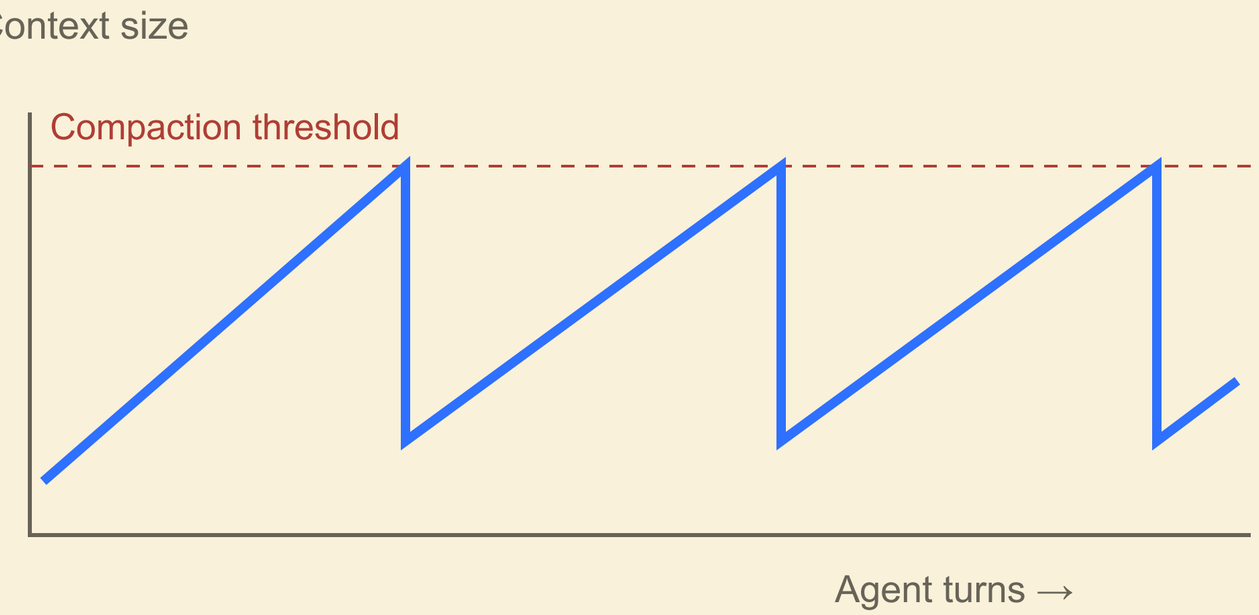
Thariq Shihipar · Anthropic · Original session-management figure · Checkpointing reference

Compaction: a summary for the next turn



Compaction on demand · Schematic

CliffCompaction keeps excerpts, then drops the old block



Keep exact fragments

+ recent turns unchanged

Drop bulky tool content

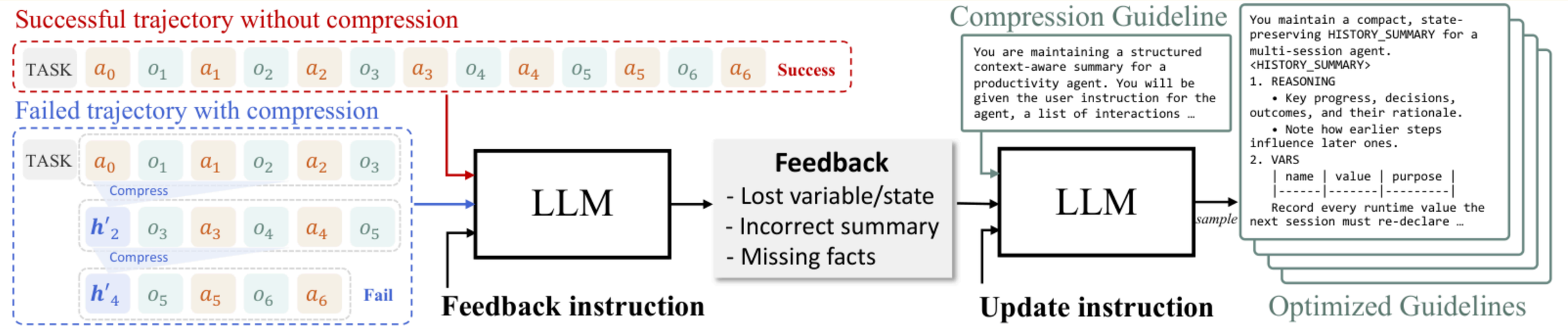
Reread files when needed

At the next cliff:

discard the old compacted block

Nguyen et al. · CliffCompaction · arXiv:2609.26779v1, §2 and Fig. 2 · September 2026

ACON learns better instructions for compressing context



Peak token usage

26–54% lower

Across the tested benchmarks

Task success

Better

Than existing compression baselines

Optimization target

Compression rules

Core method keeps agent weights fixed

Kang et al. · ACON v3 · Original Figure 3 · AppWorld, OfficeBench, Multi-objective QA · Peak tokens are not dollar savings



@rajistics

Compaction methods make different tradeoffs

Method	What changes	Teaching distinction
Summary	Rewrite older history	Risk: lose or distort a detail
ACON	Improve compression instructions	Learn what a summary must retain
CliffCompaction	Keep excerpts; drop old blocks	Risk: lose older observations

Summer gave OpenClaw a rule

1. She sets the rule

Confirm before acting

→

2. A few hours later

Emails deleted

→

3. Her explanation

Compaction lost the rule

Why? She set a rule.



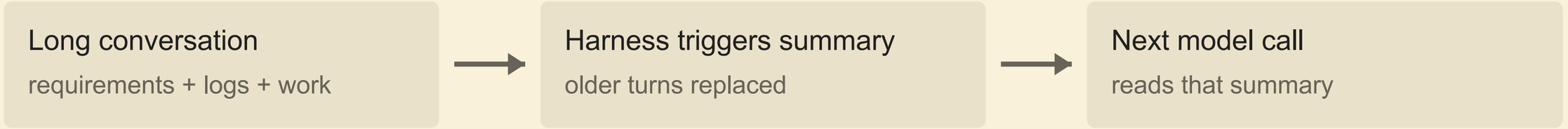
Summer Yue · Photo: SXSW 2025 profile

Summer Yue’s reported incident, February 2026.

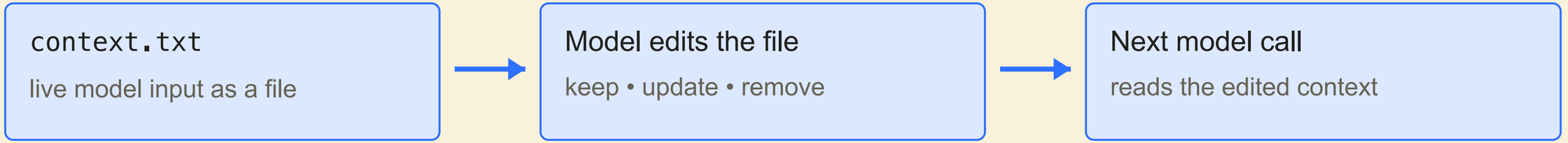
Original incident report · Yue’s explanation

Who chooses the next context?

Fixed compaction

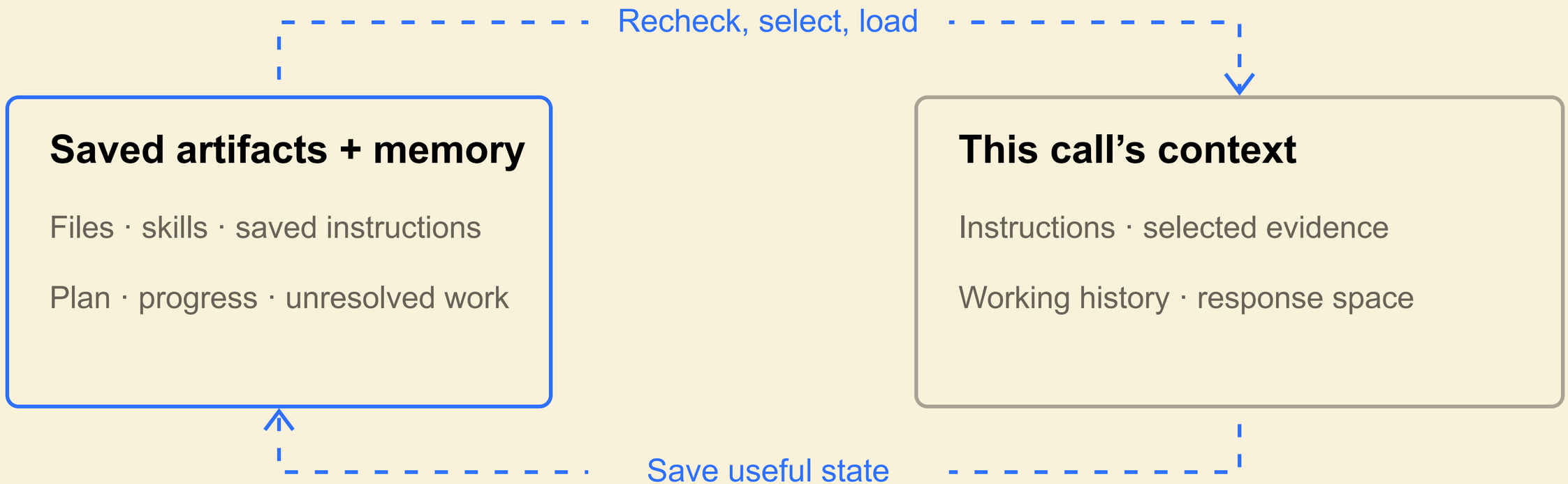


Model-controlled context



Shao et al., Context Language Models, 2026 preprint, §4.1. Diagram illustrates the mechanism; it is not a benchmark result.

Some information needs to outlive this window



✓ 1. Load

✓ 2. Compact

3. Save

4. Recheck

Task memory: save the plan, resume the work

BEFORE STOPPING

plan.md

✓ Reproduce pagination failure

✓ Find boundary calculation

□ Fix the empty-page case

□ Rerun boundary tests

NEXT SESSION

Load plan.md

Check the current files

Continue with the open step

Update the saved plan

Illustrative pagination plan · Saved state must be loaded and checked before reuse

Consider two related tasks

	SHARED SEARCH PROCEDURE	TASK-SPECIFIC FINISH
TASK 1 **Add Sony headphones to my wish list**	Go to store → Find search box → Enter query: Sony headphones → Click Search	Open Sony result → Add to wish list
TASK 2 **Find the wireless keyboard price range**	Go to store → Find search box → Enter query: Wireless keyboards → Click Search	Read result prices → Report min / max

REPRESENTATION

Text instructions, executable code, or both?

INDUCTION

How can the agent extract a procedure from experience?

LIFECYCLE

When should it retrieve, check, revise, or forget the procedure?

Adapted from Daniel Fried, CMU 11-768, Memory and Skills, p. 2



Methods for saving/updating the agent experience

LOCATION	BENEFITS	TRADEOFFS
Context window Messages and tool results	Preserves detail about what happened	Token cost and noise. Still needs relevance selection.
External artifacts Notes, skills, checkpoints	Inspectable, editable, retrievable, portable	Must be created or inferred, selected, and maintained.
Model weights Fine-tuning	Can reduce inference work. Changes behavior broadly.	Slower updates, opaque, specific to the model.

Adapted from Daniel Fried, CMU 11-768, Memory and Skills, p. 3



A map of instructions and memory

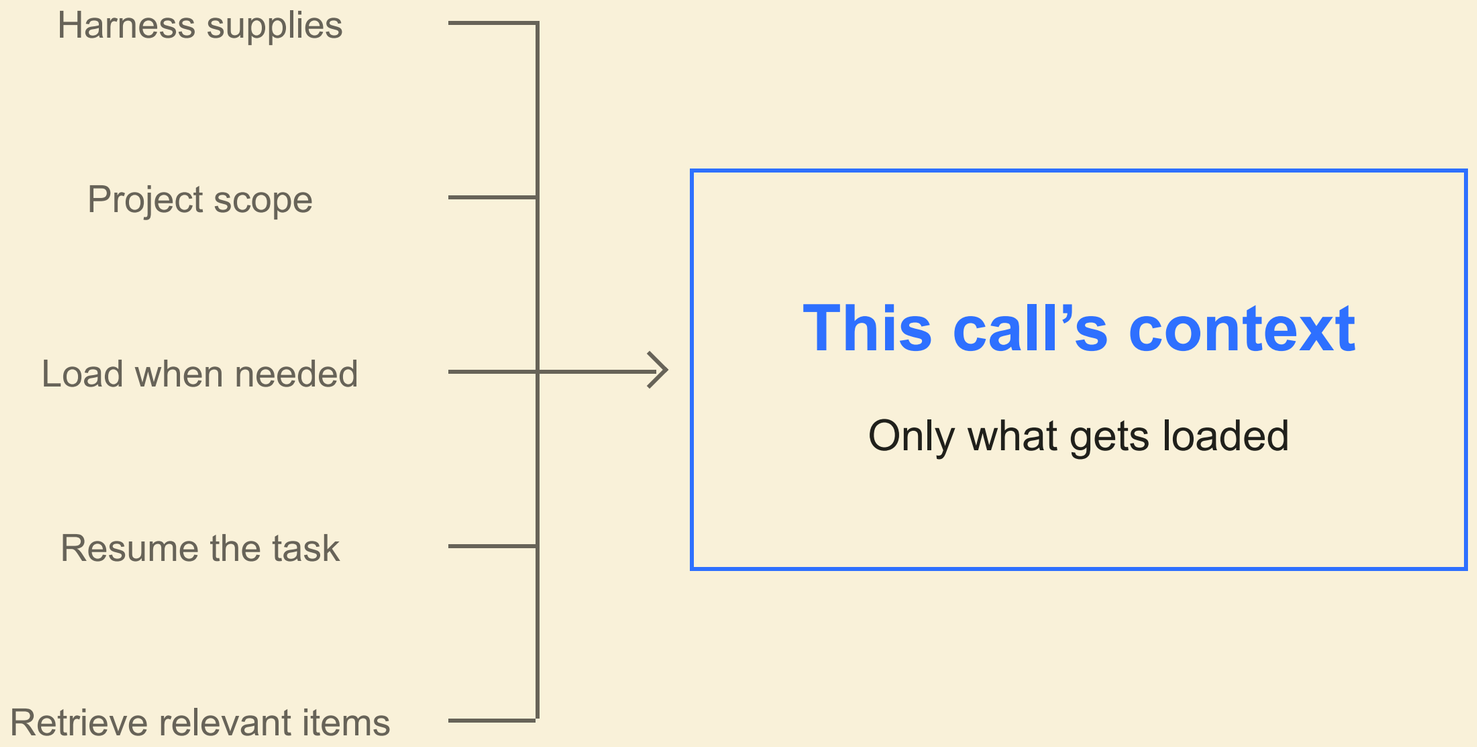
System prompt
Standing behavior + tool rules

AGENTS.md / CLAUDE.md
Project decisions + conventions

Skills
Reusable task procedures

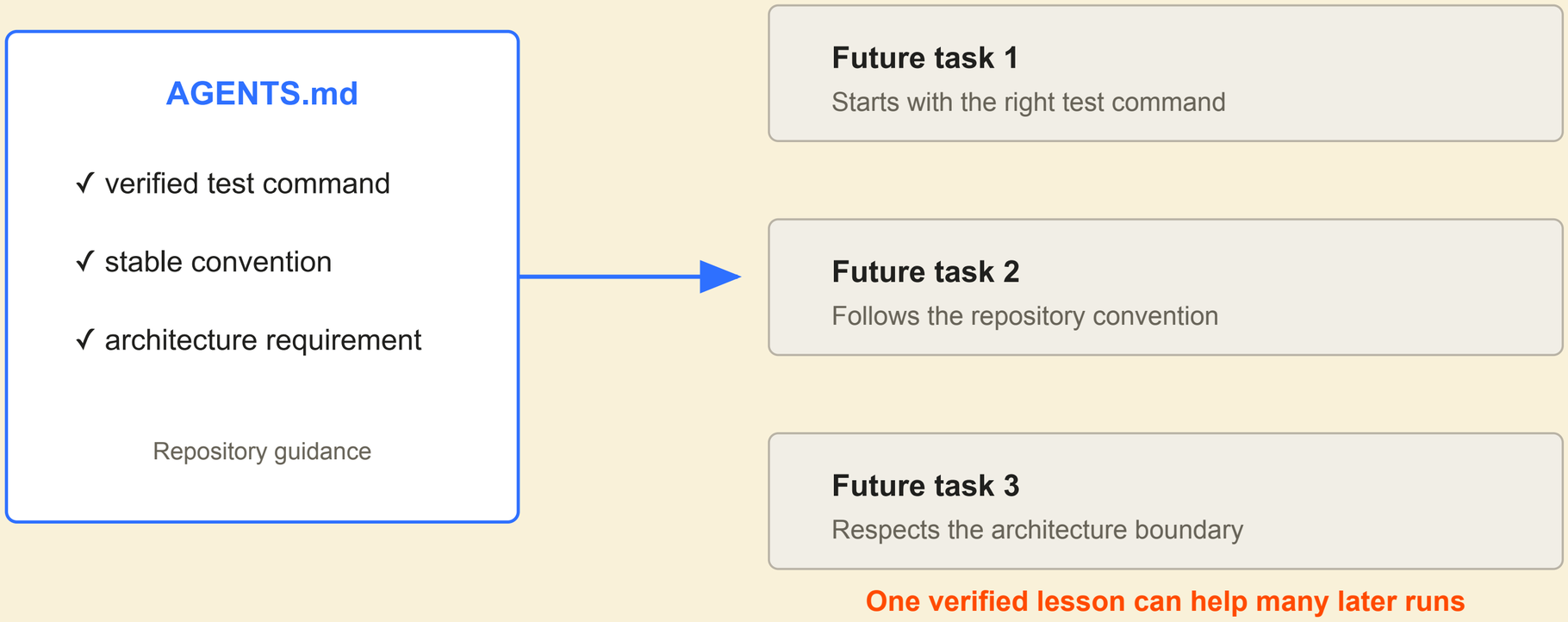
Task checkpoints
Progress + unfinished work

Memory store
Past facts + verified lessons



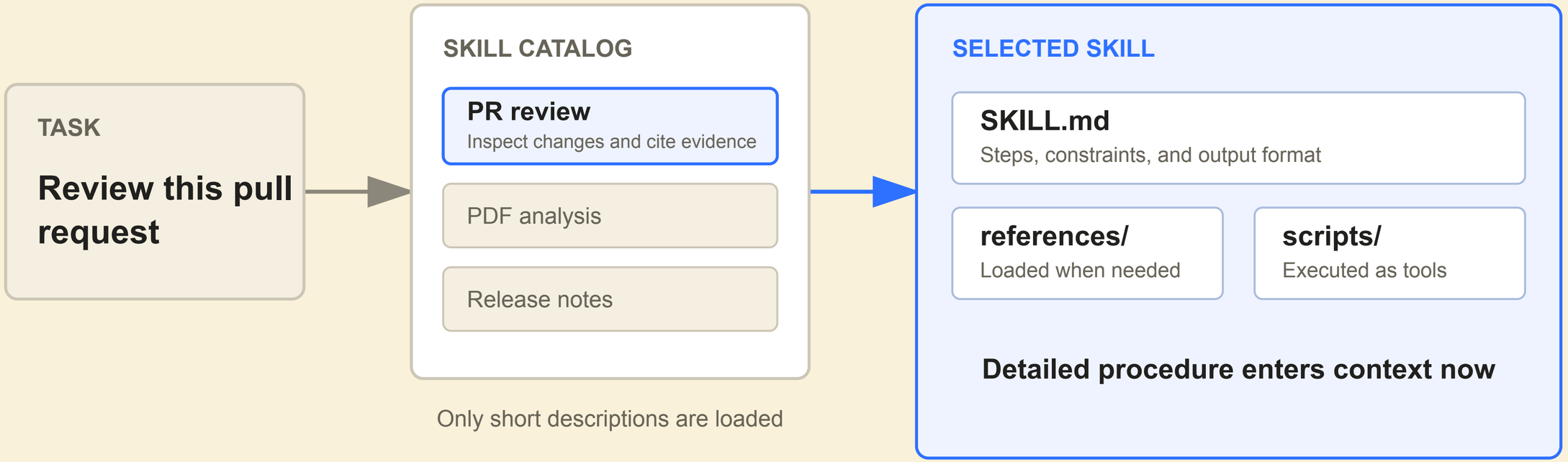
AGENTS.md · Agent Skills · Memory tool · Loading varies by harness

AGENTS.md: repository guidance for future tasks



AGENTS.md specification

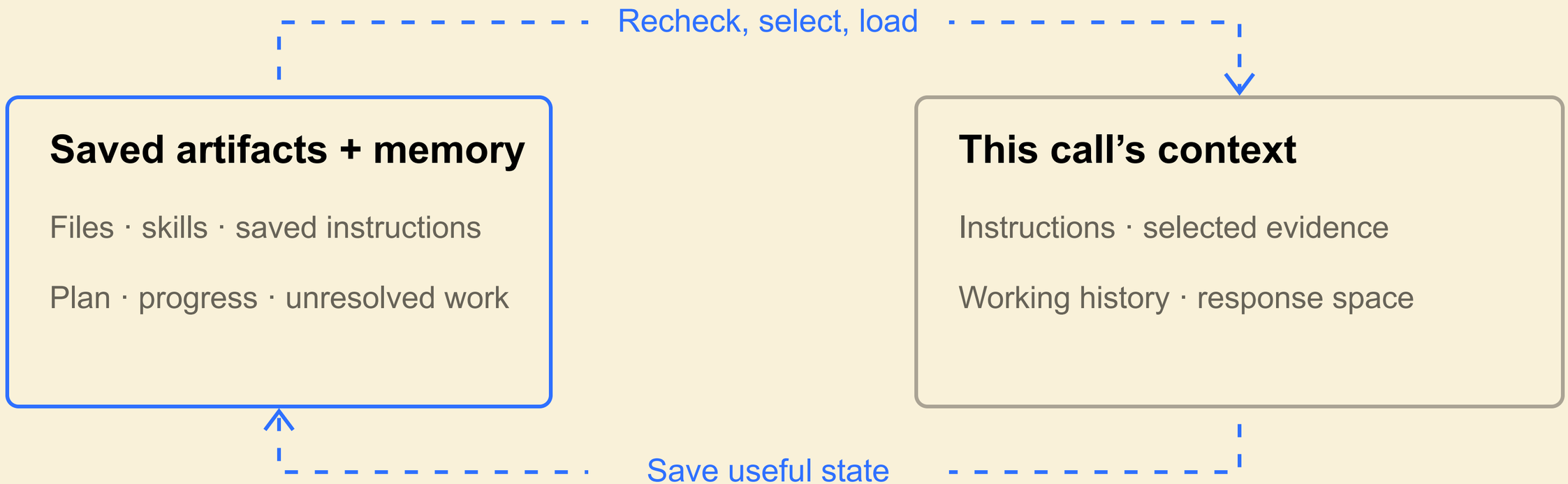
Skills only load detailed instructions when the task needs them



Progressive disclosure keeps unrelated procedures out of the prompt.

Anthropic, Equipping agents for the real world with Agent Skills

Check saved information before using it again



✓ 1. Load

✓ 2. Compact

✓ 3. Save

4. Recheck

Workshop 3: Effect of memory

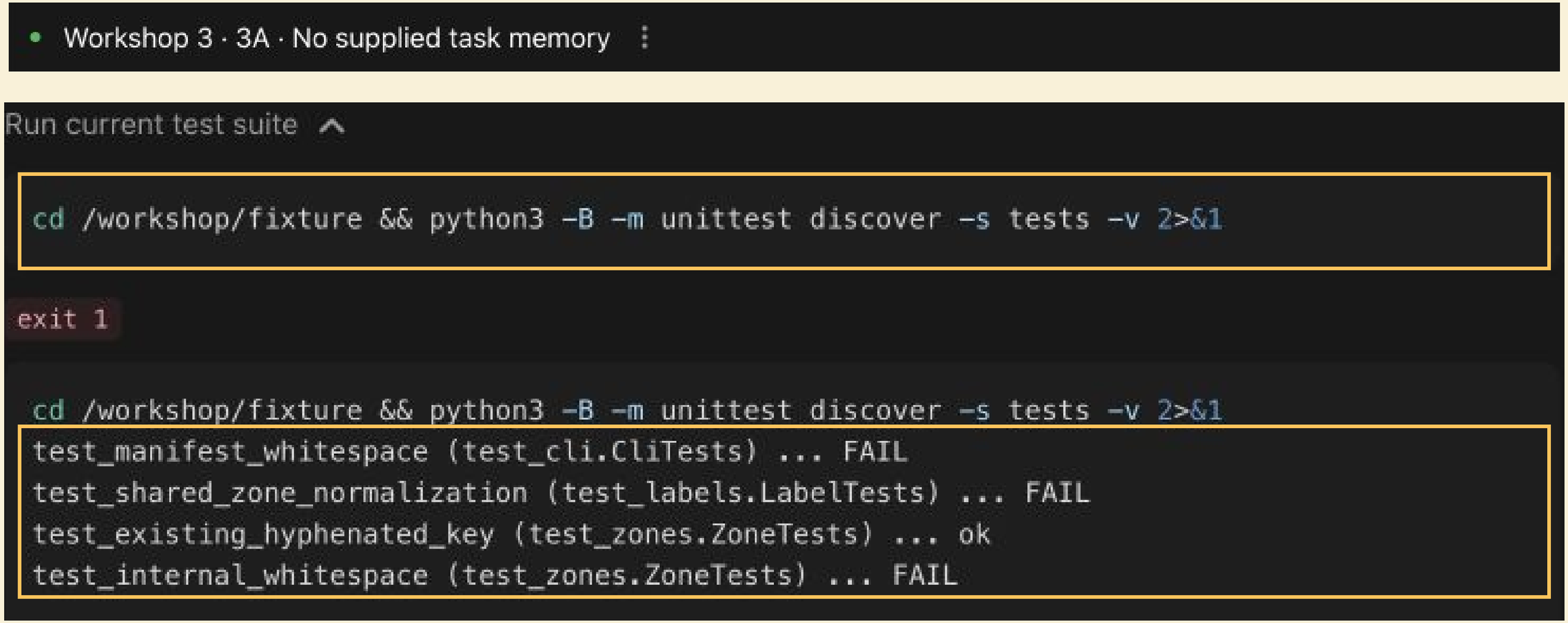
1. Save a memory note → **2. Give it to the next run** → **3. Compare the first check**

Did it use it?

Compare no memory, memory, and stale memory

Same task, fixture and model · One fresh pair · Supplied note, not automatic retrieval

No memory: the first check runs the full suite



First check: 4 cases, 3 failures. The complete history includes the later repair.

3A · No supplied task memory · Complete history · Paths/user normalized in archive

Useful memory supplies a focused check

Workshop 3 · 3B · Useful saved route

SUPPLIED

routekit/cli.py:main calls routekit/manifest.py:build_manifest, which delegates zone keys to routekit/zones.py:canonical_zone. routekit/labels.py:zone_label uses the same helper. Focused regression: python3 -B -m unittest -v tests.test_zones. Full suite: python3 -B -m unittest discover -s tests -v. This is a navigation map, not a patch or a prior passing verdict. Source files and test command were inspected/executed; recheck current code and tests. Evidence: l02-l06.

FIRST CHECK

```
cd /workshop/fixture && python3 -B -m unittest -v tests.test_zones 2>&1; echo "exit=$?"
```

RETURNED

```
Ran 2 tests in 0.000s

FAILED (failures=1)
exit=1
```

Focused check omits CLI and label tests. The final full suite passes all 4.

3B · Saved route supplied in prompt · Complete history · Paths/user normalized in archive

Stale memory points to a missing check

Workshop 3 · 3C · Stale saved route (earlier take)

SUPPLIED

```
Saved check command to try first: python3 -B -m unittest -v tests.test_zone_smoke
```

USED

```
Run saved smoke check command
```

RETURNED

```
        "type": "text",
        "text": "{\"behavior_cases_executed\": 0, \"command\": \"python3 -B -m unittest -v te
      }
    ]
```

```
\"error\": \"ModuleNotFoundError: tests.test_zone_smoke\", \"exit_status\"
```

The stale route runs 0 behavior cases.

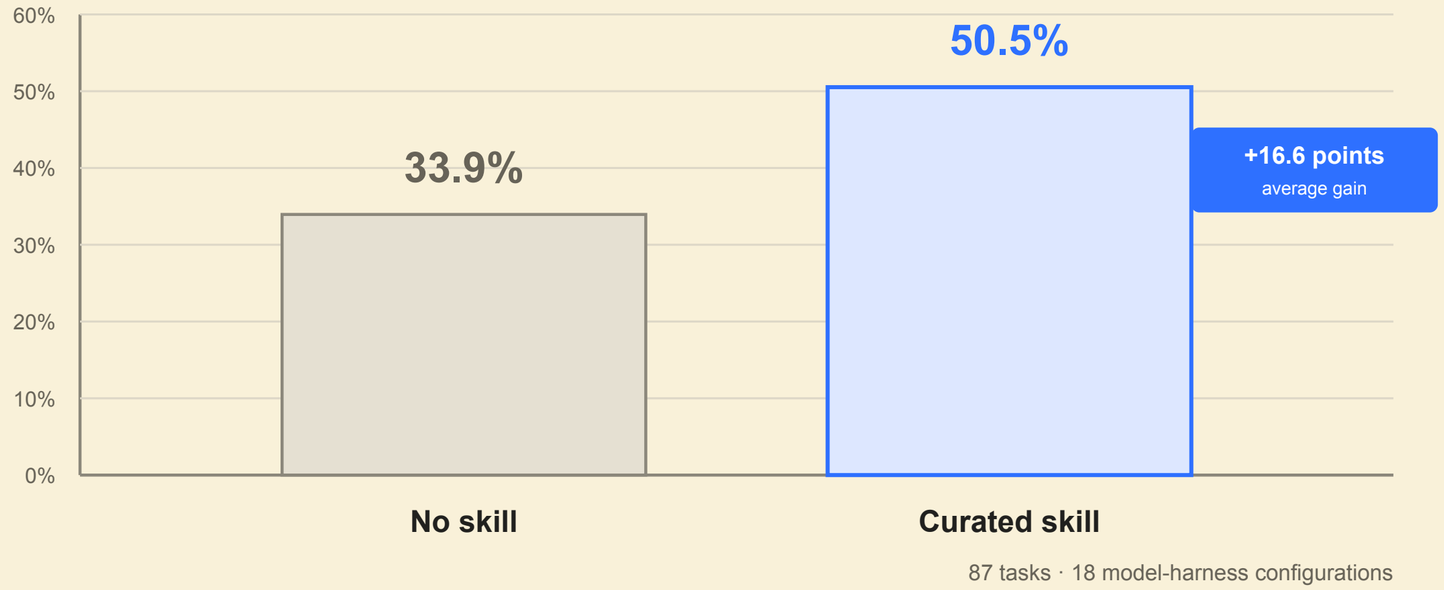
3C · Earlier take · Custom tools · Reuse-first prompt · Complete history

Workshop 3: memory comparison

CONDITION	FIRST CHECK	FIRST RESULT	FINAL RESULT
3A · No memory	Full suite	4 cases 3 failures	**4 / 4 pass**
3B · Useful memory	Saved focused check	2 cases 1 failure	**4 / 4 pass**
3C · Stale memory	Obsolete check	0 behavior cases Loader error	**4 / 4 pass** After recovery

3A/3B: fresh matched pair · 3C: earlier custom-tool take; reuse required · No speedup demonstrated

Skills help on average, but some tasks get worse



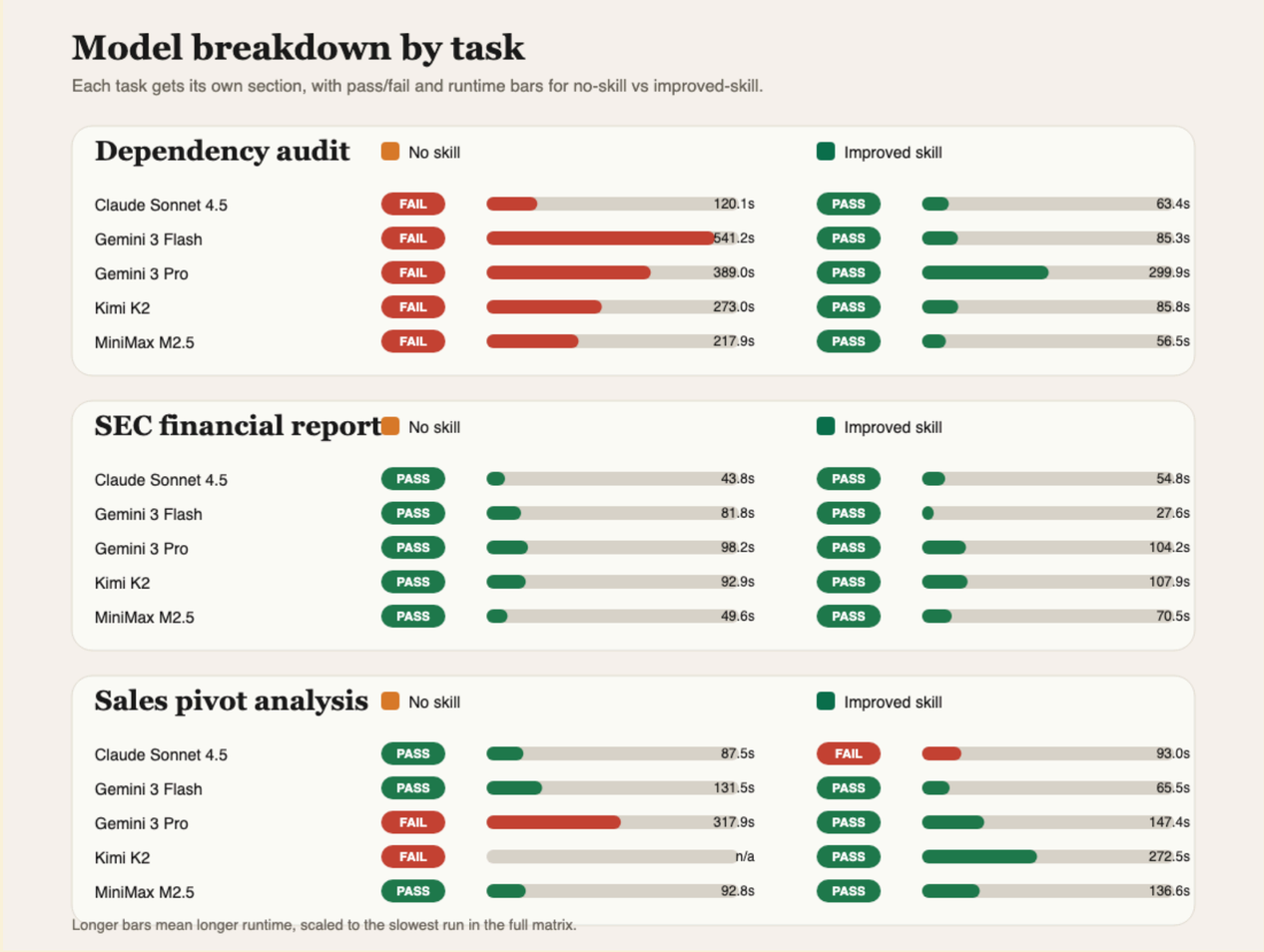
13 of 87

tasks had lower pass rates with skills

Test the skill on its intended work

SkillsBench v4 · Average across 87 tasks and 18 configurations · Negative task deltas: §5.1.3

Evaluate a skill on the work it should improve



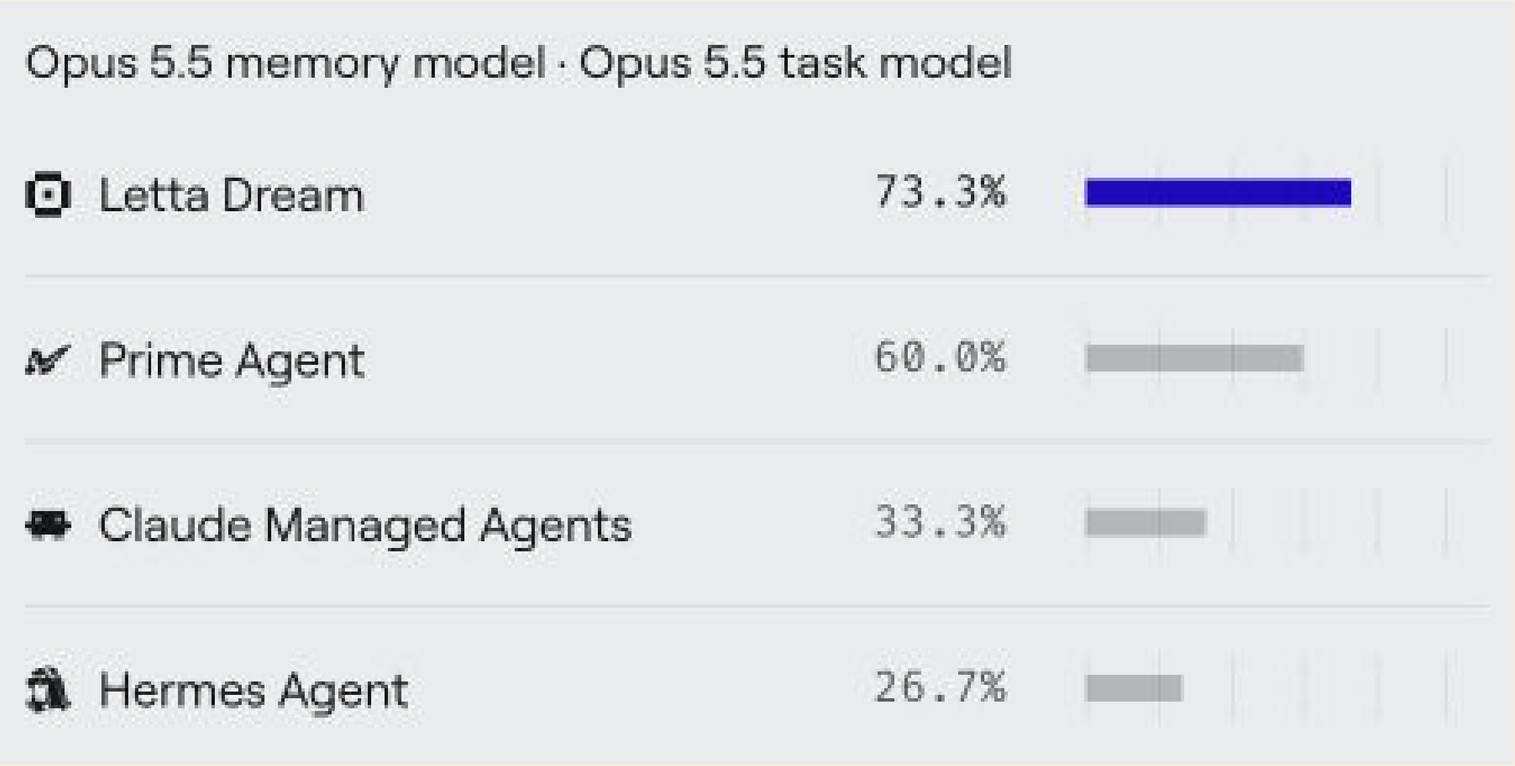
- Compare the same work with and without the skill.
- An average gain can hide failures on individual tasks.

Blog post: https://openhands.dev/blog/evaluating-agent-skills
Repo: https://github.com/rajshah4/evaluating-skills-tutorial;



@rajistics

FrontierMemory tests memory on later tasks



Illustrative earlier feedback

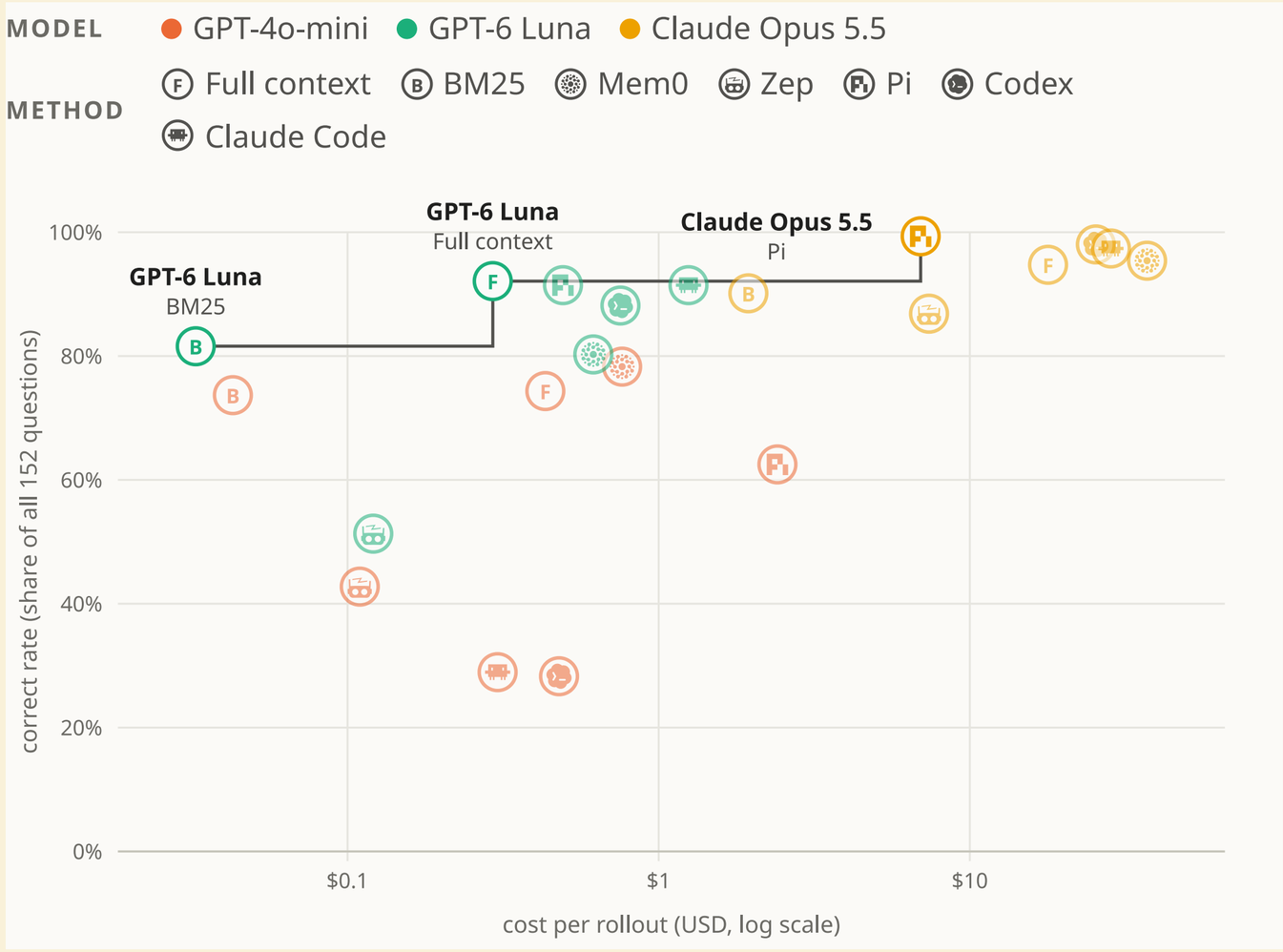
“Changelog: one line, newest first.”

Later task

Fix a bug and write the changelog.

Letta Research · FrontierMemory 1.0 · Original Figure 2, Opus 5.5 panel

Do we need a memory layer, or better retrieval?



Raw-history search

Pi + Opus 5.5

99% · $6.95

Full context · Opus 5.5

95% · $17.84

Photo access differed

Peng · MemoryTax v1 · Figure 1 · One conversation, 152 questions, one run/cell · Costs exclude judging

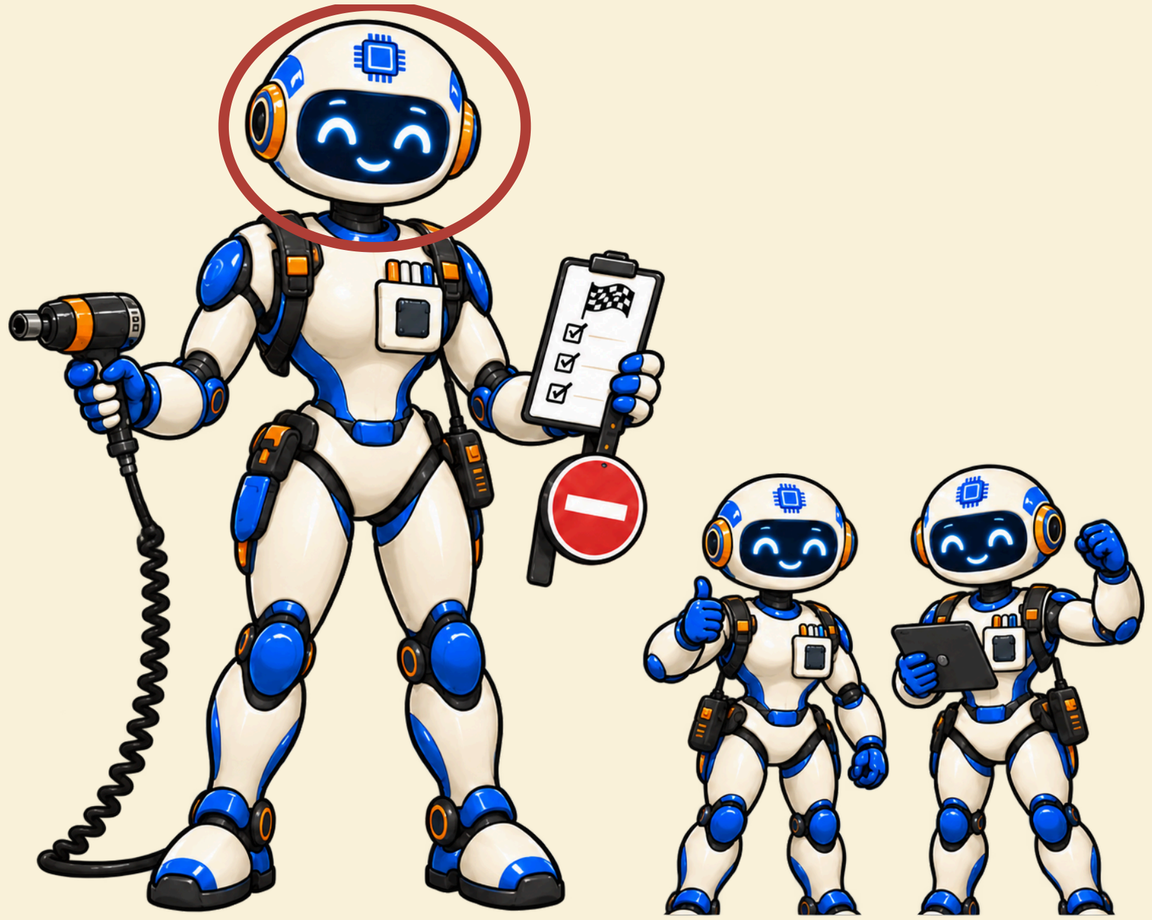
Context and memory: four design decisions

1. Load	Retrieve and maintain high quality context
2. Compact	Compact as history grows, preserving instructions and goals.
3. Save	Use checkpoints and external memory artifacts like AGENTS.md and skills.
4. Recheck	Learn from past runs. Check that saved guidance still helps the current task.

SECTION

IV

Model routing



Workshop 4: When should we use a stronger model?

04

- Start with Flash, the lower-cost baseline.
- Compare Pro, the stronger-model candidate.
- Use the same checks to decide whether to escalate.

Workshop exercises

is.gd/odsc2026_harness

Exact shared SVG prompt

Let's compare creating an animated SVG

Same prompt: a hedgehog plays an accordion, paws attached, body swaying, seamless loop.

DeepSeek V4 Flash



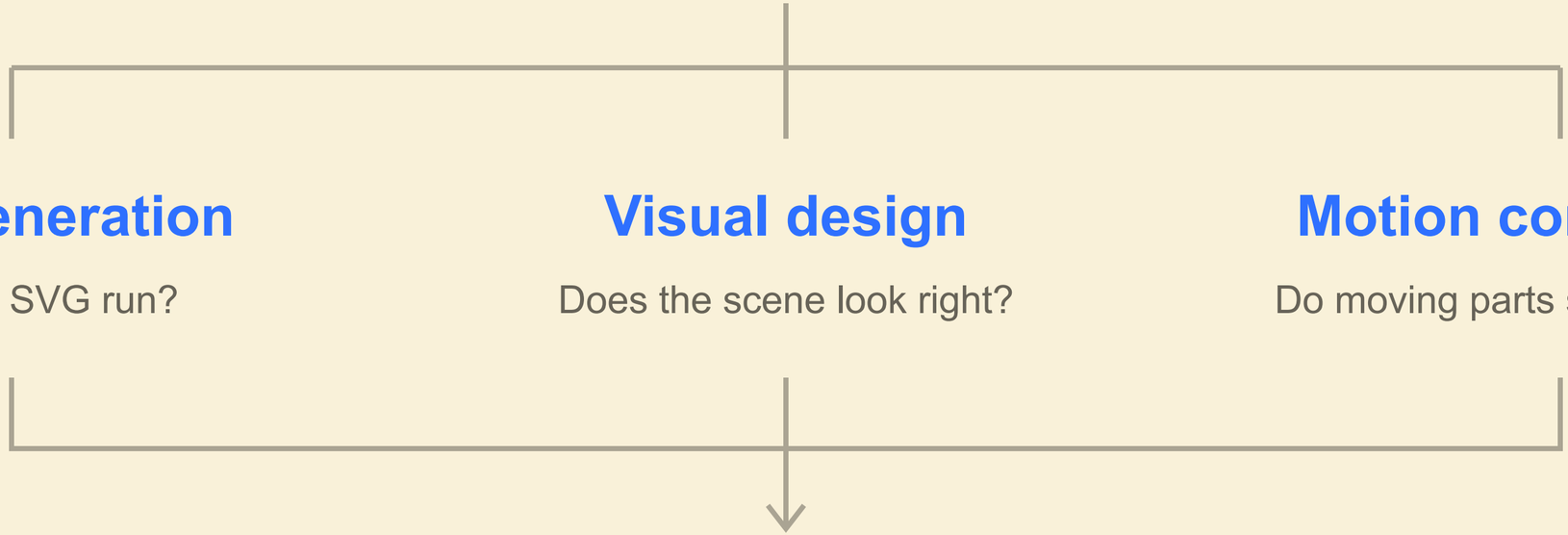
DeepSeek V4 Pro



One first output per model · identical prompt and 16,384-token limit · no retries

Models have different strengths

One animation task. Several abilities.



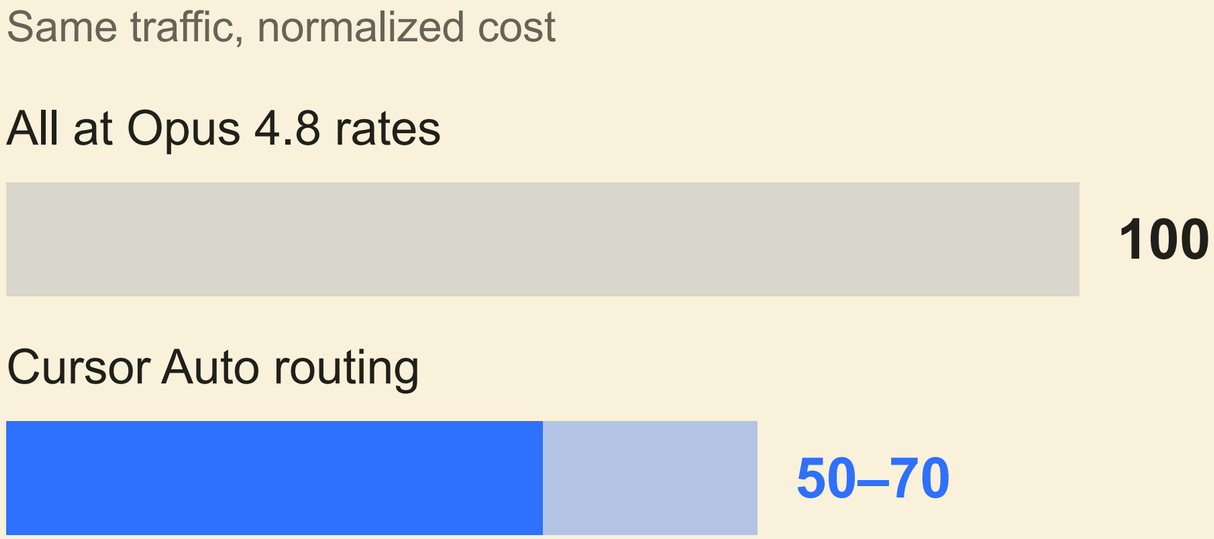
Match the model to the task.

One output per model illustrates the differences; it does not establish a general ranking.

Cursor adds model routing

30–50%

lower model cost



Cursor, Introducing Cursor Router · July 22, 2026. Vendor-reported early-access results; chart normalizes the reported range.

Maximizing model routing

You should change models at every step of a task

Switching models can trade away cache reuse

Stay with A

Reuse the eligible cached prefix

Same model + matching prefix + provider cache rules

New input

Switch A → B

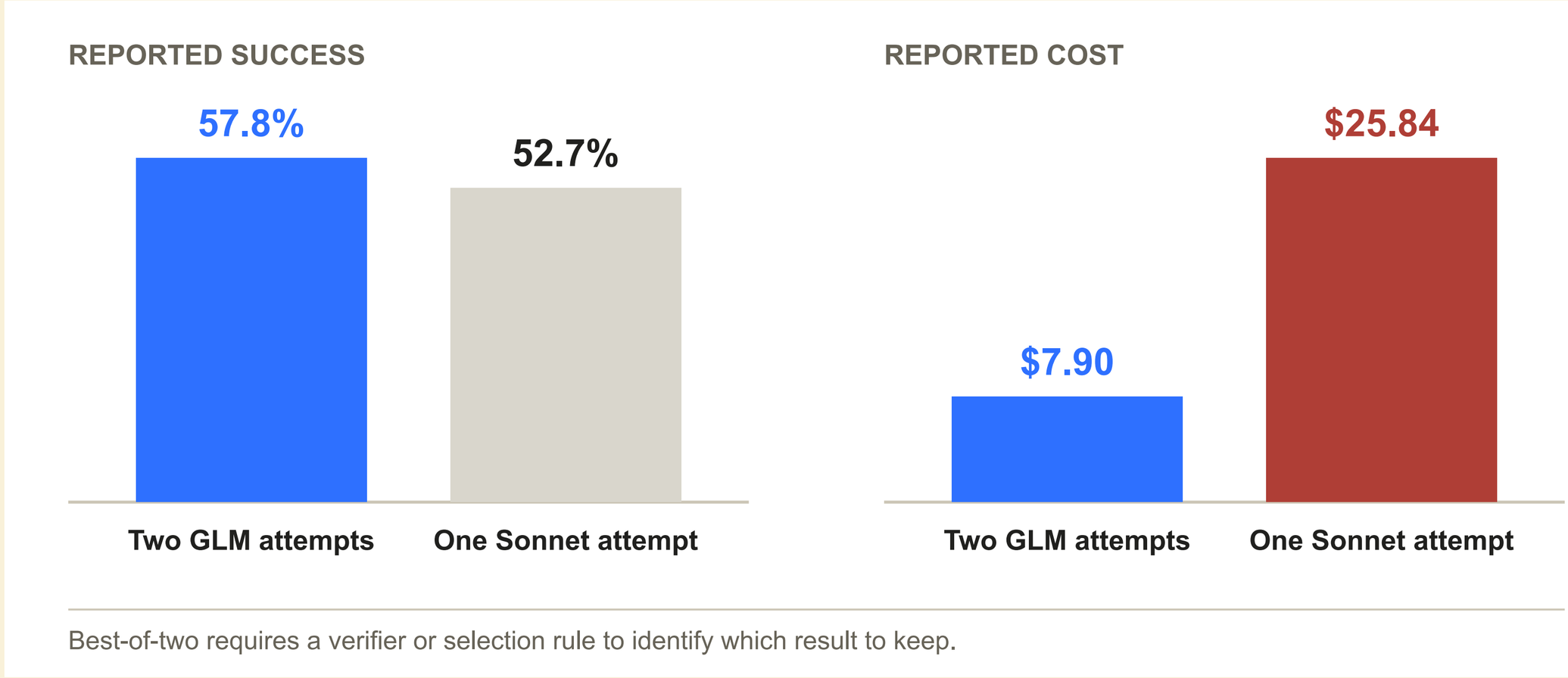
Process the history again

Model B does not inherit model A's cache

New input

Anthropic, prompt caching in Claude Code

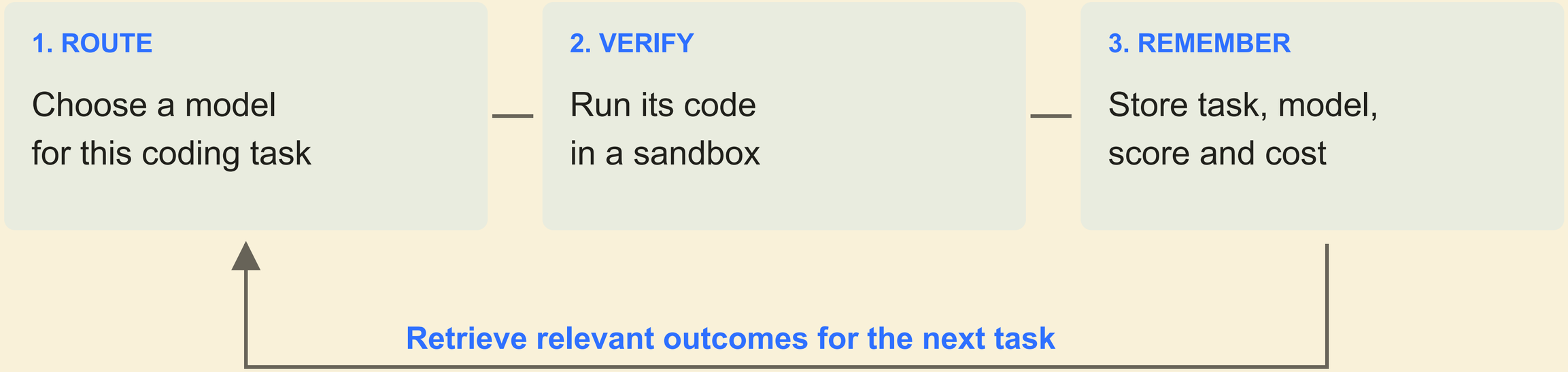
A retry can compete with a pricier model



Best-of-two requires a verifier or selection rule to identify which result to keep.

Zain Hasan, DeepSWE rollout analysis. 452 rollouts per model. Best-of-two requires a way to select the winner.

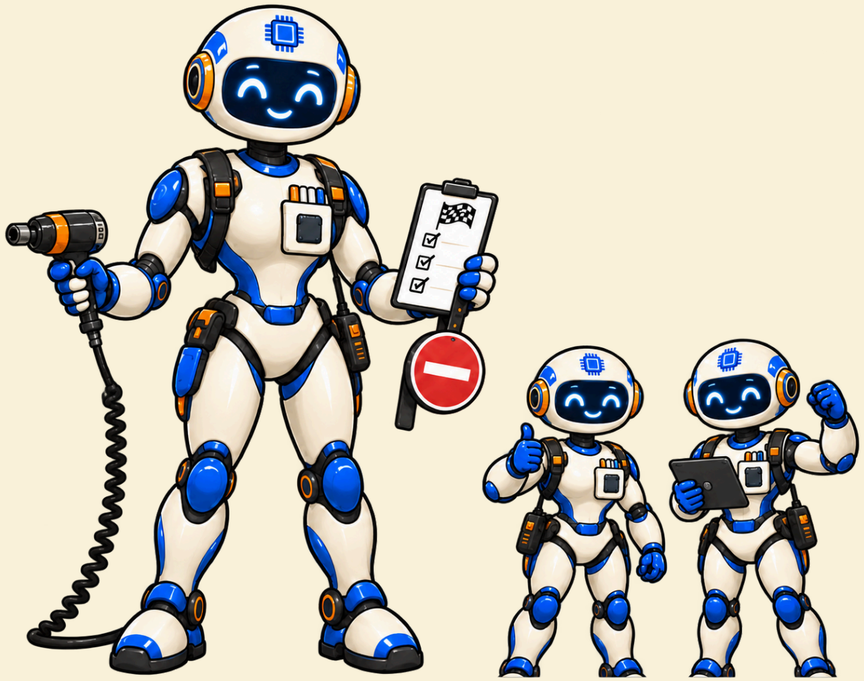
Choose the next model using observed results



The feedback changes the router’s available evidence.

Zhou et al. · Agent-as-a-Router, §3.3 / Figure 2 · Process redrawn from the paper

Choosing a different model versus training a model



Choose an alternative model

Match its abilities to the task.

Weights stay unchanged.

Train for repeated work

Update weights using examples or rewards.

Evaluate with the intended harness.

Model choice and routing: takeaways

01 Match model capabilities to task requirements.

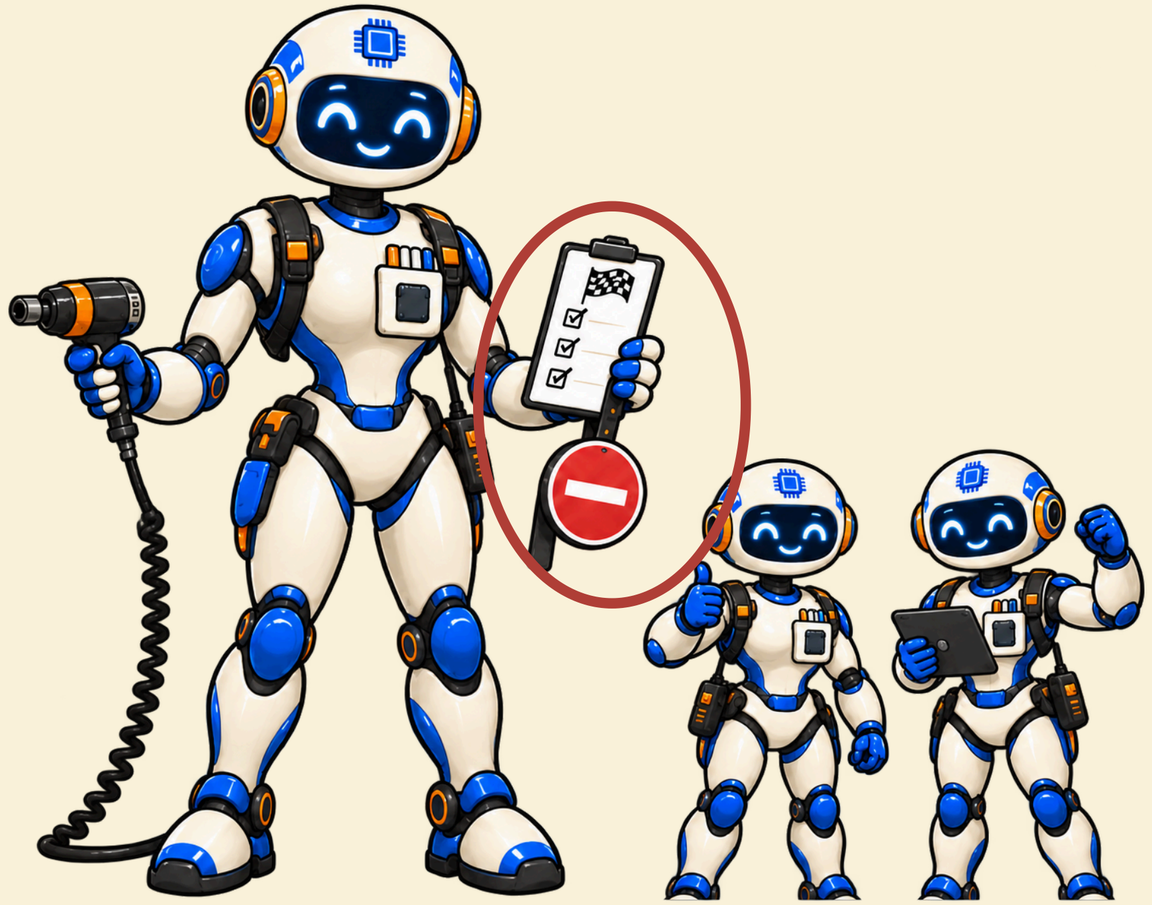
02 Use acceptance checks to guide retries and escalation.

03 Optimize the cost of successful work.

SECTION

V

Goals and stopping



A 35-hour run without accepted progress

Task: Python with virtual threads and lexical scoping.

Reported run measure	Result
Wall-clock time	**35 hours**
Token use	**1B / 4B in source**
Raw API cost	**About $1,200**
Repository activity	**79 commits, 75K net lines**
Armin's assessment	**“Absolutely nothing of value”**

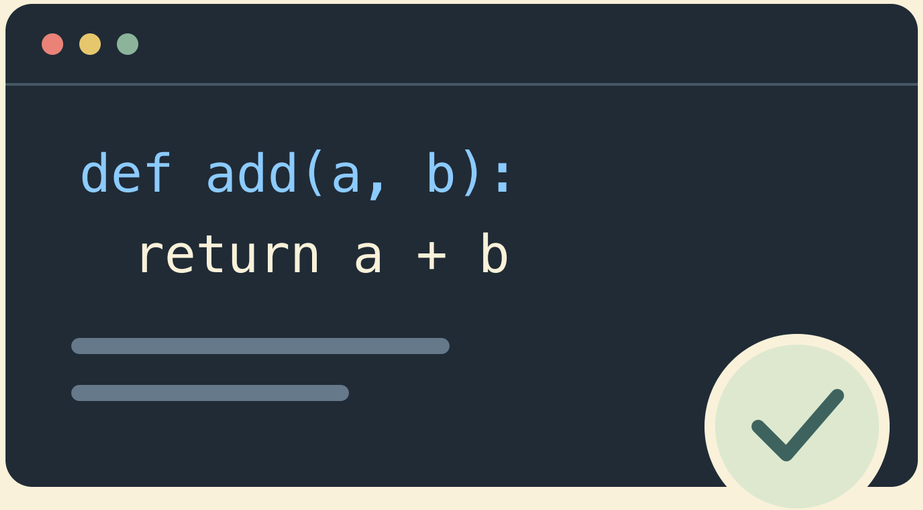
Source gives conflicting token estimates. Armin stopped the unattended run himself.

Armin Ronacher, Astra for Coding: Why Are We Doing This Again?



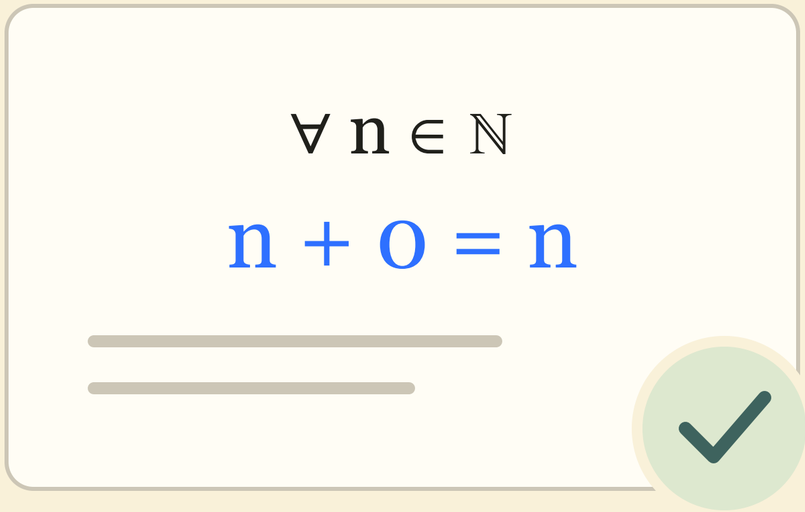
Verifiable outcomes give learning a signal

Code



Test cases

Math

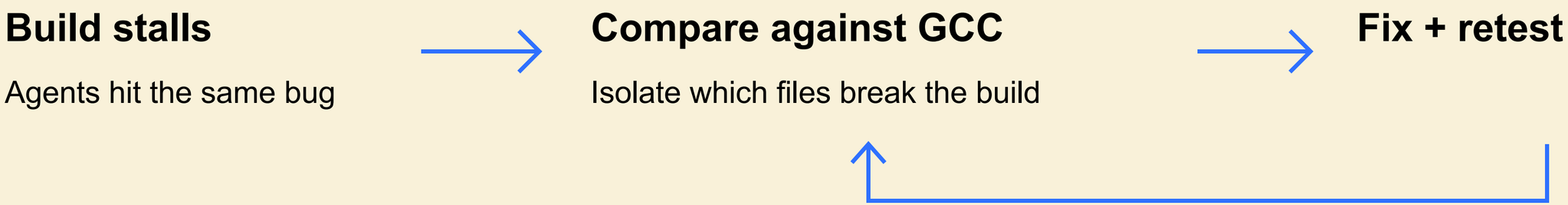


Lean verifier

DeepSeek-R1, §2.2 · AlphaProof · Illustrative visuals

How GCC helped

Goal: build a C compiler capable of compiling Linux.



REPORTED OUTCOME

Built Linux 6.9

16 agents · ~2,000 sessions

~$20,000 · two weeks

Nicholas Carlini, Anthropic · Building a C compiler with a team of parallel Claudes · Feb. 5, 2026

Build verification into the agent’s control loop

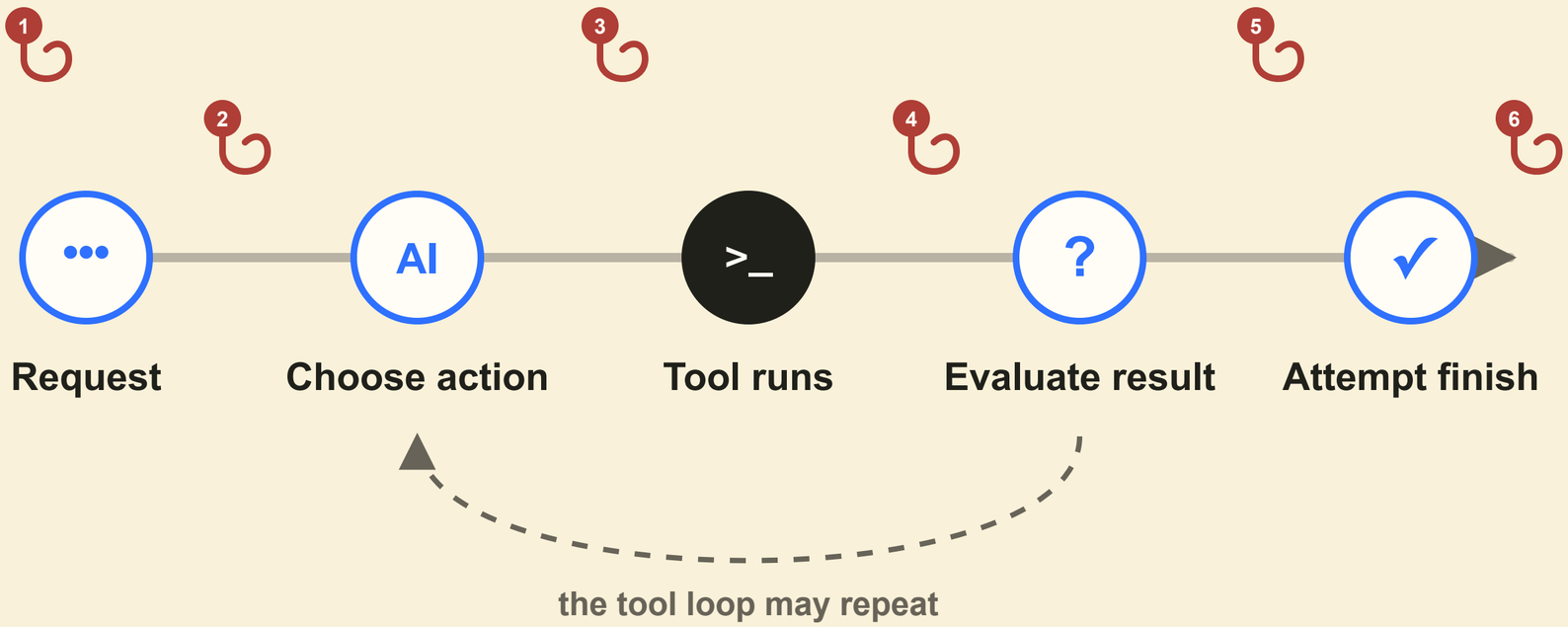
GOAL → REQUIRED EVIDENCE → CHECK → NEXT ACTION

EXECUTABLE CHECKS	**CI passes · clean install succeeds** Example: install the release candidate in a clean environment.
JUDGMENT CHECKS	**Release notes match the actual changes** Example: a reviewer checks the user-facing explanation.
CONTROLLER	**Return the failed check, or stop at the limit** Example: missing install log → request it before approval.

Workshop design synthesis · Executable evidence, judgment, and bounded continuation are distinct responsibilities.

Hooks run deterministic logic at defined moments

AGENT TRAJECTORY



deterministic hook

OpenHands hook points

1	**SessionStart**	begin
2	**UserPromptSubmit**	prompt
3	**PreToolUse**	before tool
4	**PostToolUse**	after tool
5	**Stop**	finish
6	**SessionEnd**	close

Make “prepare a release” checkable

Outcome	A checked candidate, ready for review.
Evidence	CI, clean install, accurate release notes.
Boundary	Work in a branch. Do not publish yet.
Stop rule	Two repairs or ten minutes, then report.

Illustrative release policy. Set limits and approval boundaries for your own work.

Workshop 5: Verify outside the agent

Source + prepared report

Report says “complete.”

One record is missing.



External verifier

Compare IDs + total.

Precheck + Stop check.

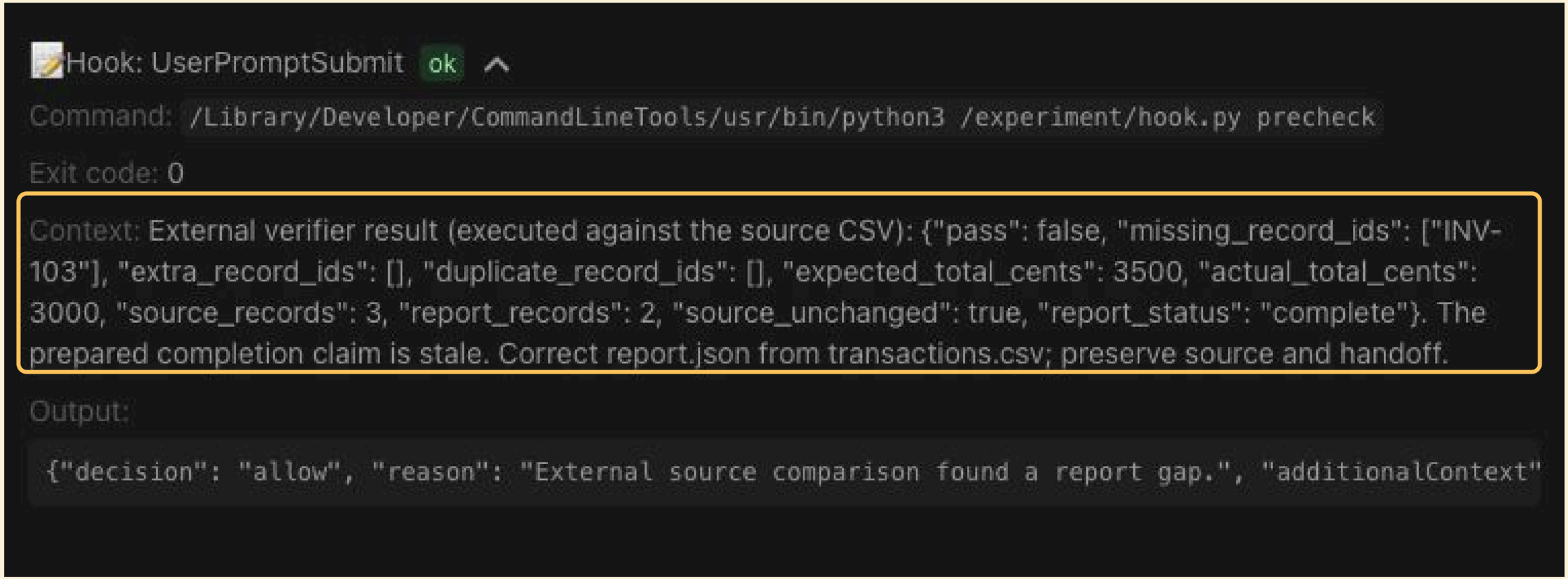


Agent repairs

Edit the report.

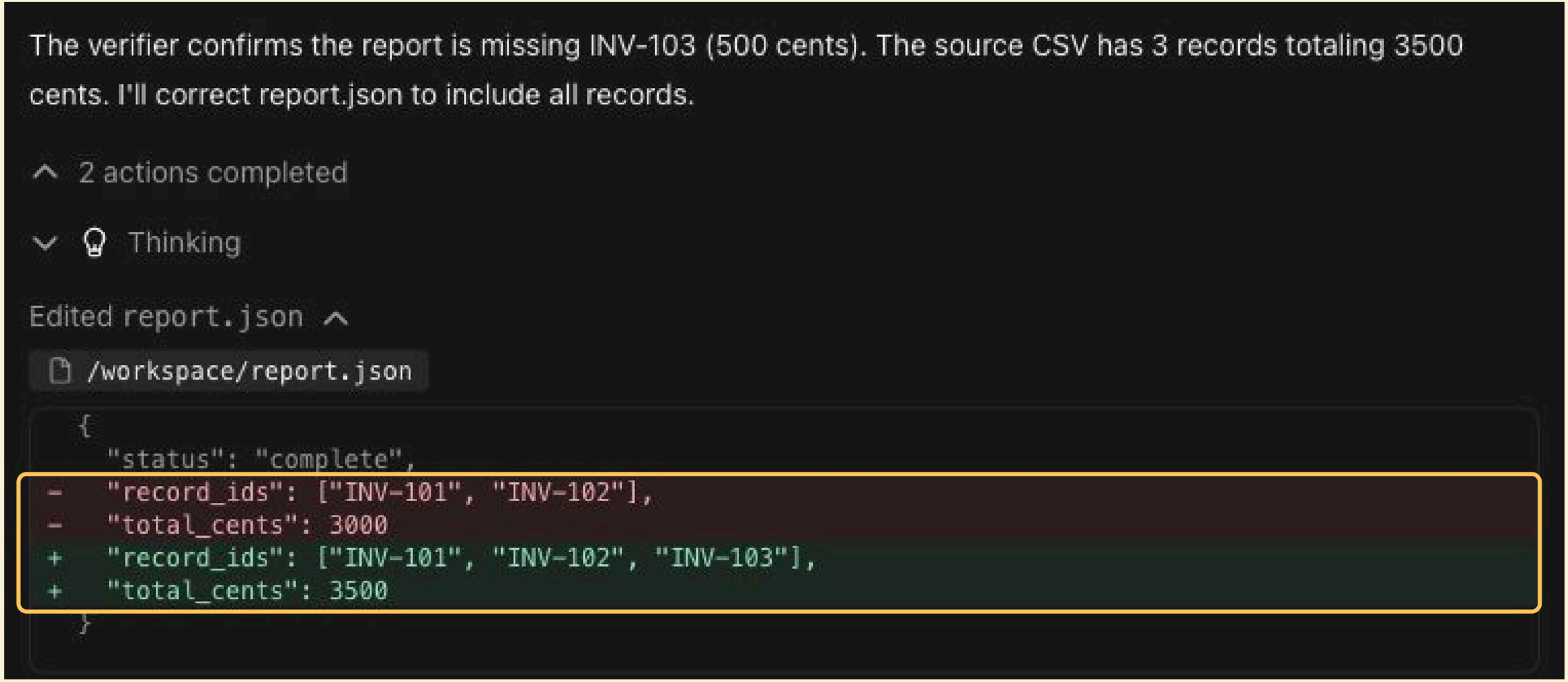
Respond to the discrepancy.

The external check finds the missing record



Hook “ok” means feedback delivered. The report check says “pass”: false.

Concrete feedback becomes one repair



Same run · adds INV-103 and changes the total from 3,000 to 3,500 cents.

The same external check now passes



The first Stop check passes. No rejected finish or automatic retry occurred.

How a harness can support verification

Define success from the source

All three records.
Total: 3,500.

Return actionable failures

Missing INV-103.
Total is 500 short.

Check the repair

All three records included?
Total equals 3,500?
Yes: task is done.

Recorded demonstration: prepared gap, explicit repair prompt, independent recheck.

Goals define success. Plans choose the next steps.

GOAL · What must be true

Make the report match the source records and total.

Plan

Reproduce the failure.
Find the cause. Choose a fix.

Act

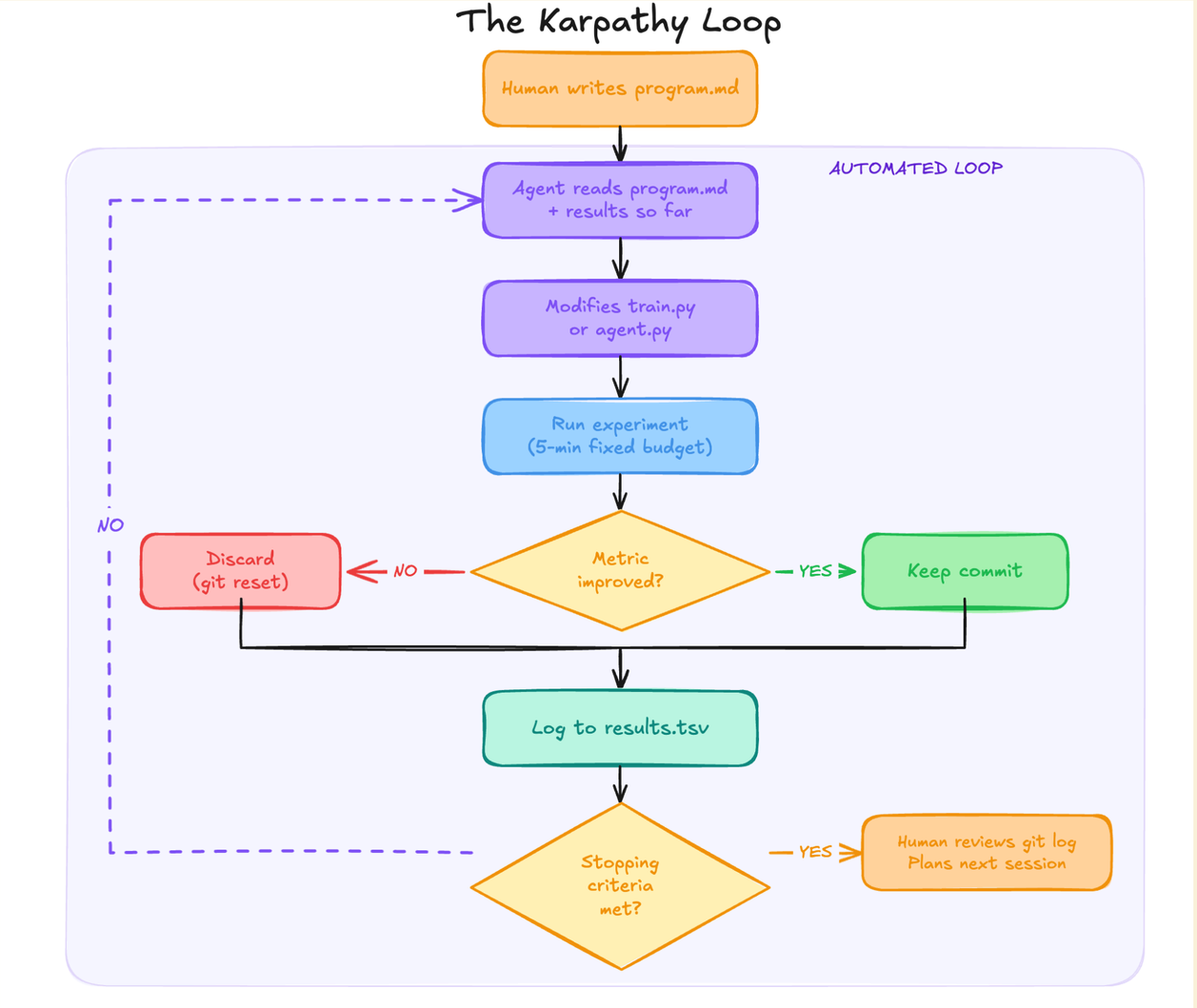
Make the next change.

Check

IDs and total match source?
Do existing checks pass?

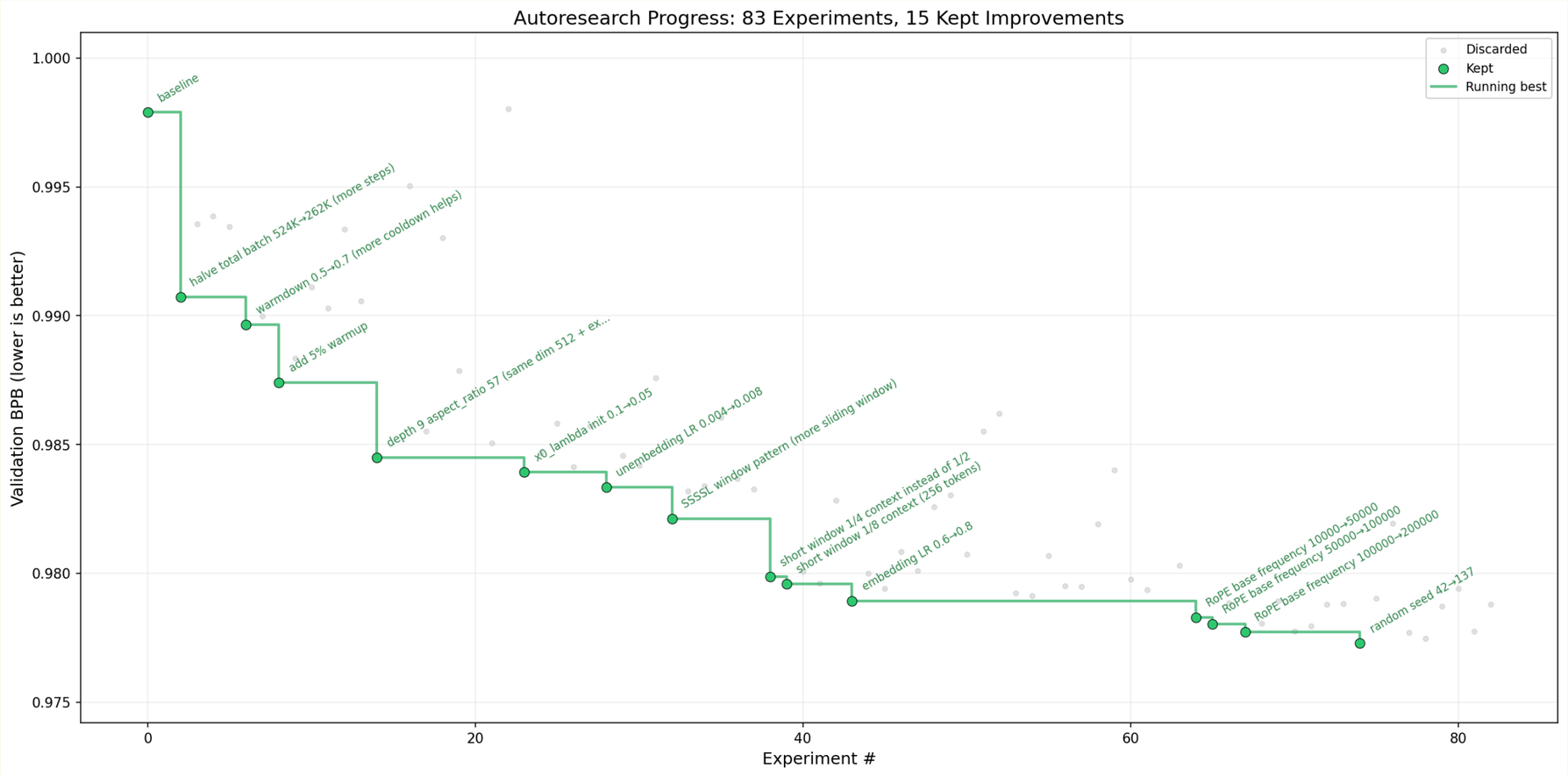
Unfinished? Use the evidence to revise the plan.

AutoResearch keeps only improvements



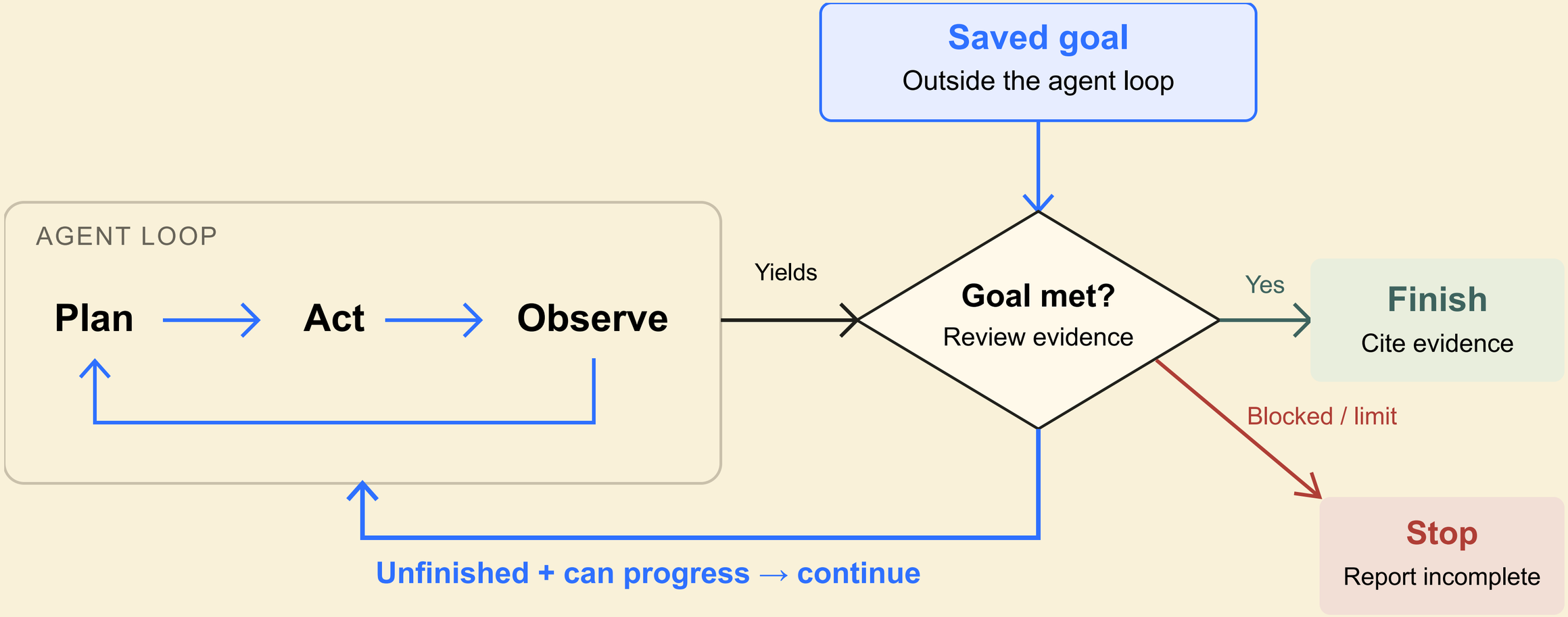
AutoResearch experiment loop · Karpathy project

AutoResearch: 83 experiments, 15 kept improvements

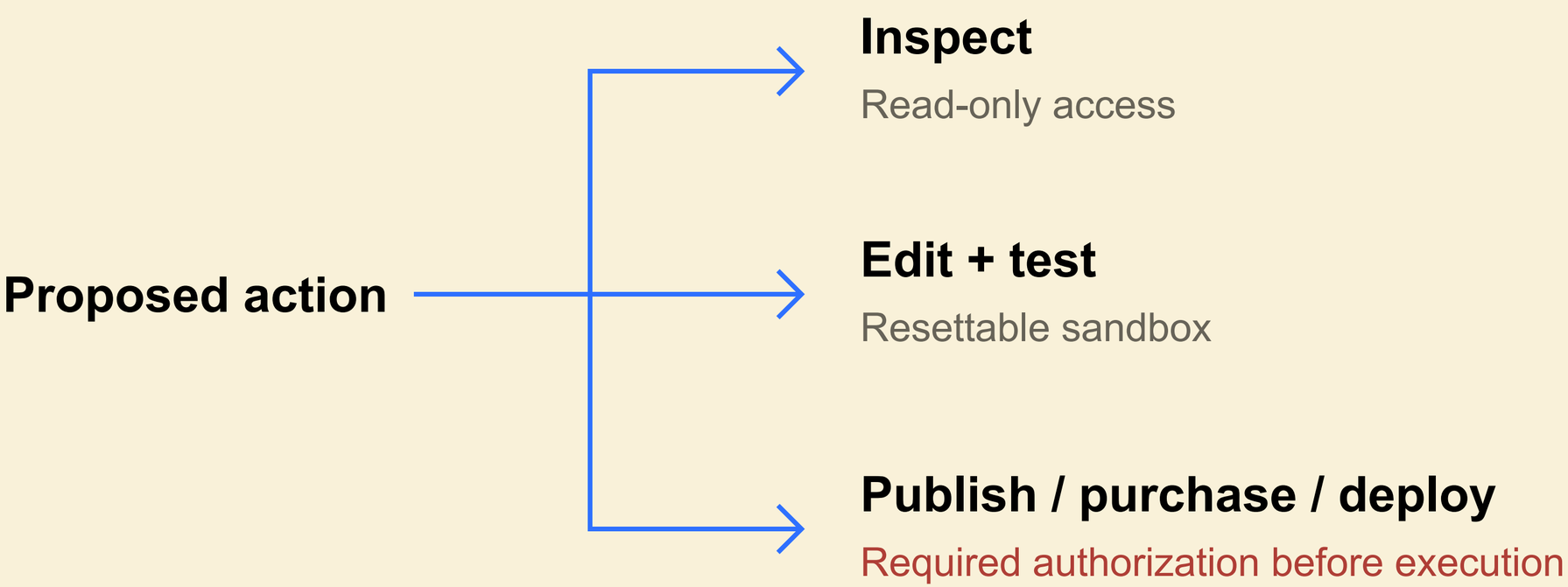


Karpathy, AutoResearch. Original progress.png; one reported run.

An external goal can restart an unfinished task

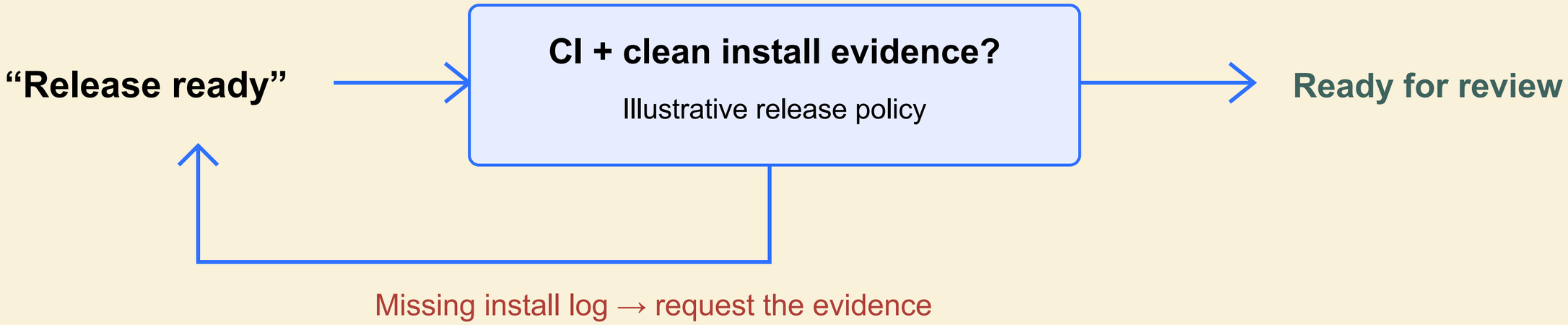


Before acting: choose the execution boundary



Anthropic, permission controls · CMU 11-768, Lecture 7

Require evidence before accepting “done”



Unsupported stops: 40 / 66 → 0 / 66

Liu · Evidence-Carrying Termination · arXiv:2608.23623v1 · Illustrative diagram

Goals and stopping: takeaways

01 Define success and constraints. Let the plan change.

02 Use independent checks to guide repair and keep improvements.

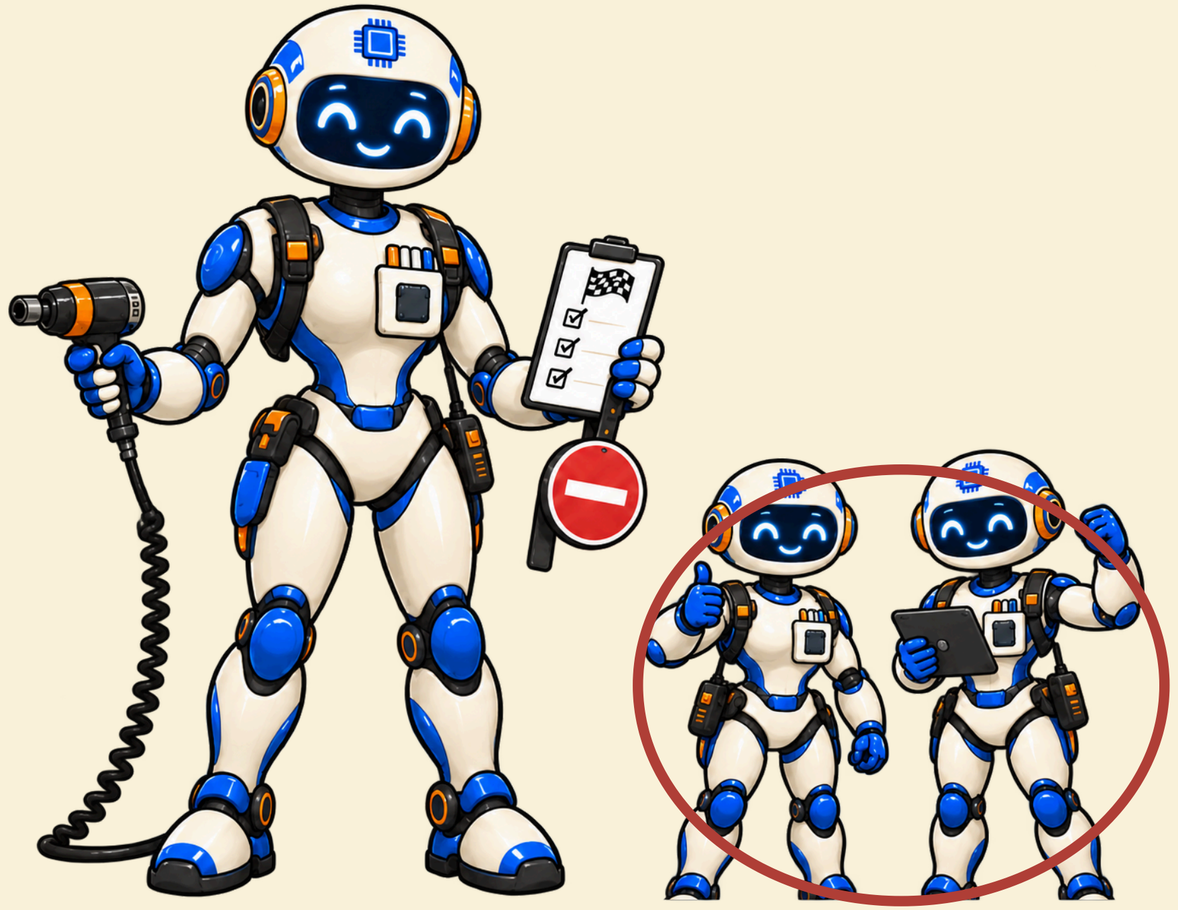
03 Require evidence for done. Check actions as well as output.

04 Continue with a useful next step. Stop at success, a limit, or a blocker.

SECTION

VI

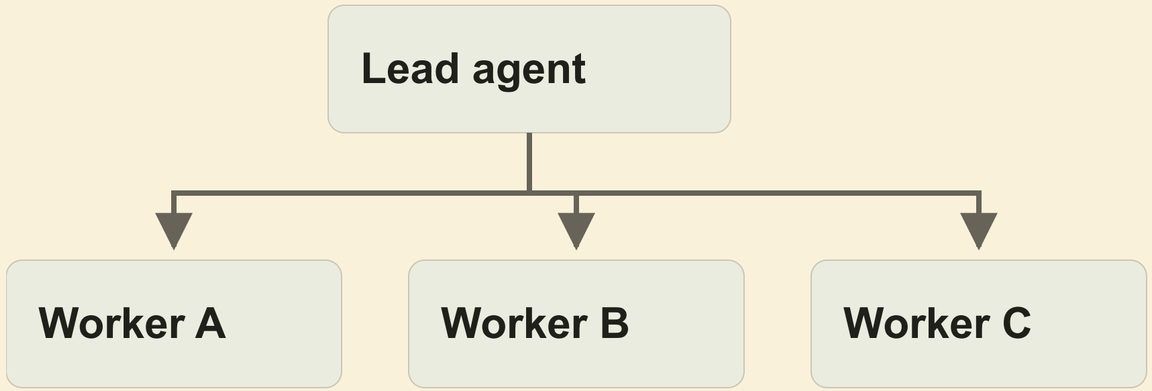
Multiple agents



Multi-agent means dividing work across agent contexts

Separate agent contexts, coordinated toward one goal.

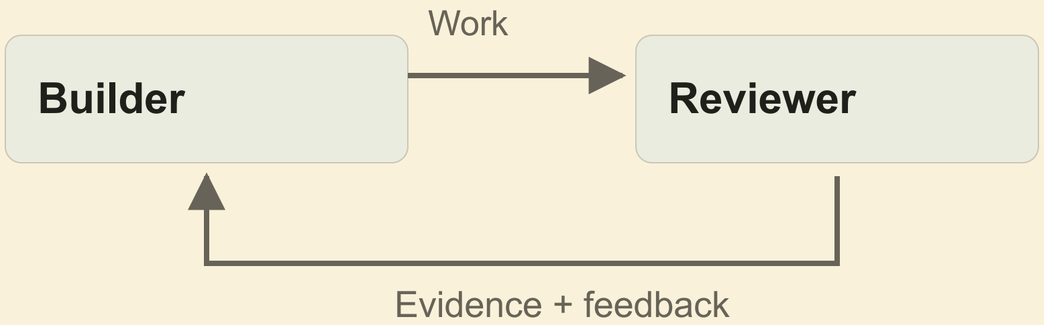
LEAD + WORKERS



Delegate pieces → return findings → integrate

Example: investigate separate parts of a codebase.

BUILDER + REVIEWER



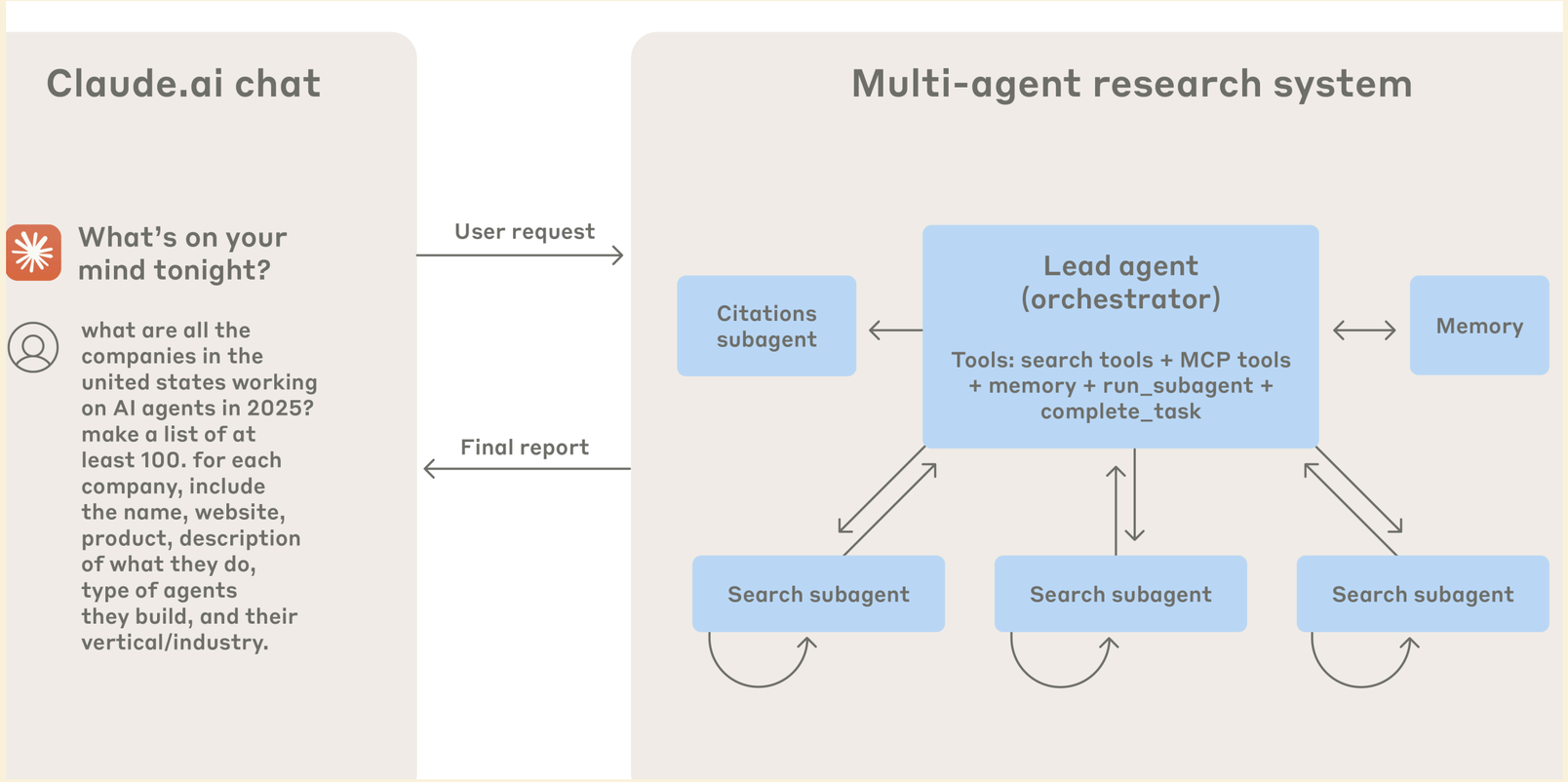
Produce → check → revise

Example: repair code, then independently probe it.

Same model is possible. The design choice is how work and evidence move.

Adapted from Anthropic: Building effective agents · Multi-agent research system

Multi-agent research scored 90.2% higher in Anthropic's eval



+90.2%

vs single Opus 4
Internal research evaluation
Opus 4 lead
+ Sonnet 4 workers

Quality/performance gain, not speed. Raw scores and exact aggregation were not published.

Anthropic · How we built our multi-agent research system · June 13, 2025



@rajistics

Workshop 6: One agent versus a team

06

One agent or a lead with three workers

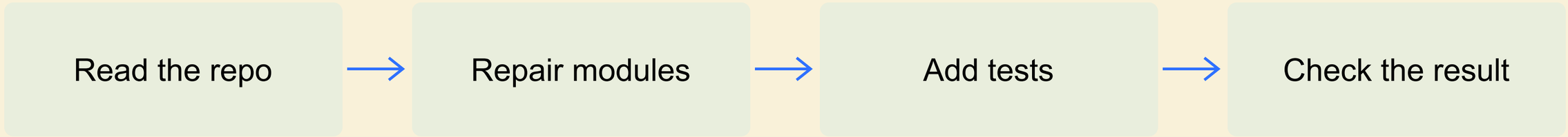
Same repair, starter and Flash model

When did it work? How much did it cost?

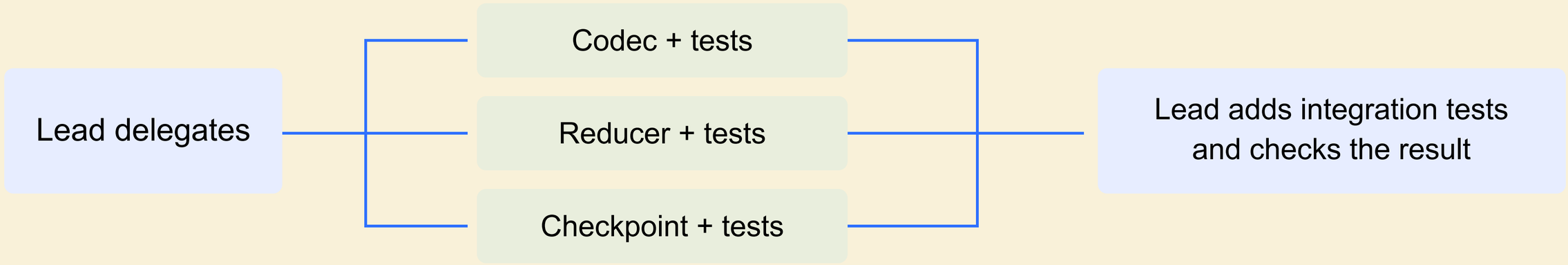
Open all five recorded conversations

One owner, or a lead and three workers

Single owner

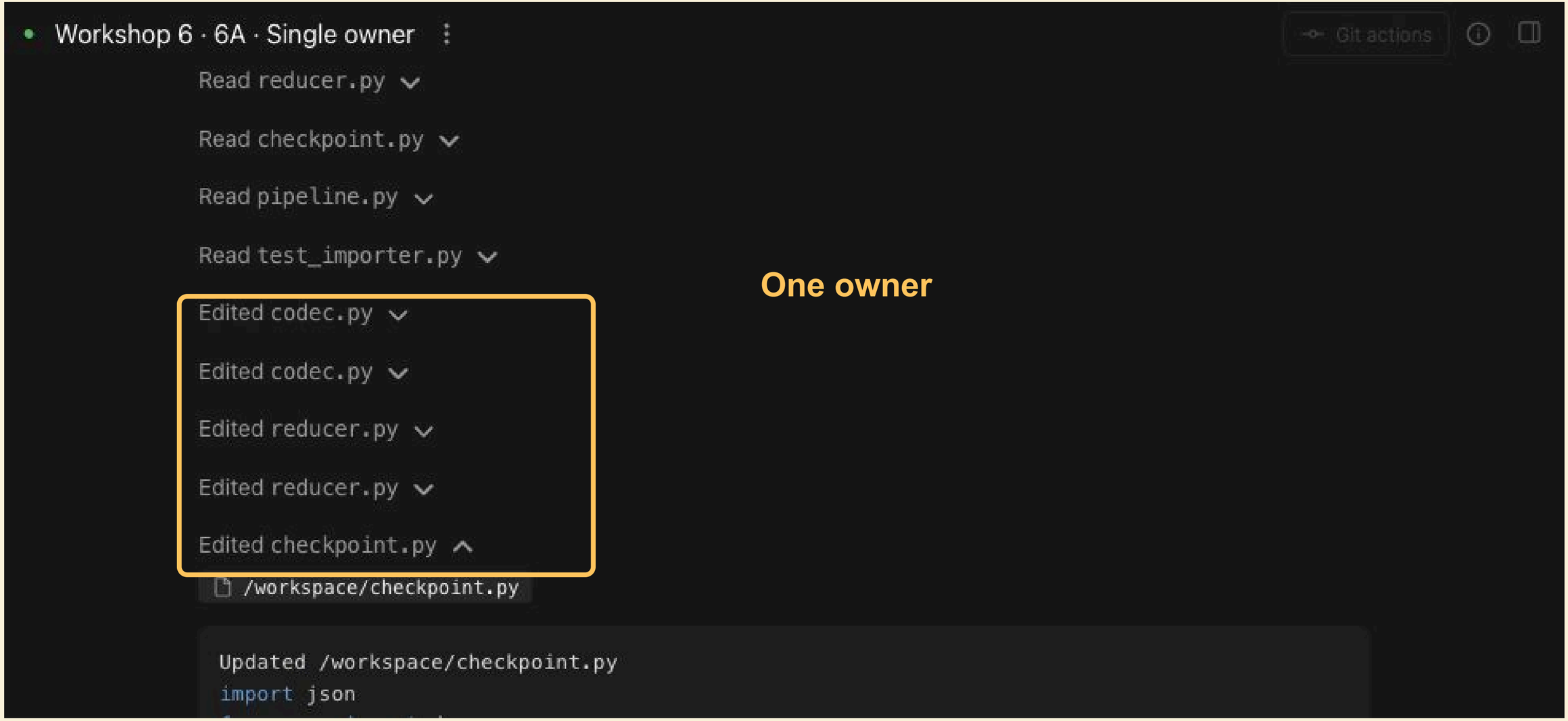


Lead + three parallel workers

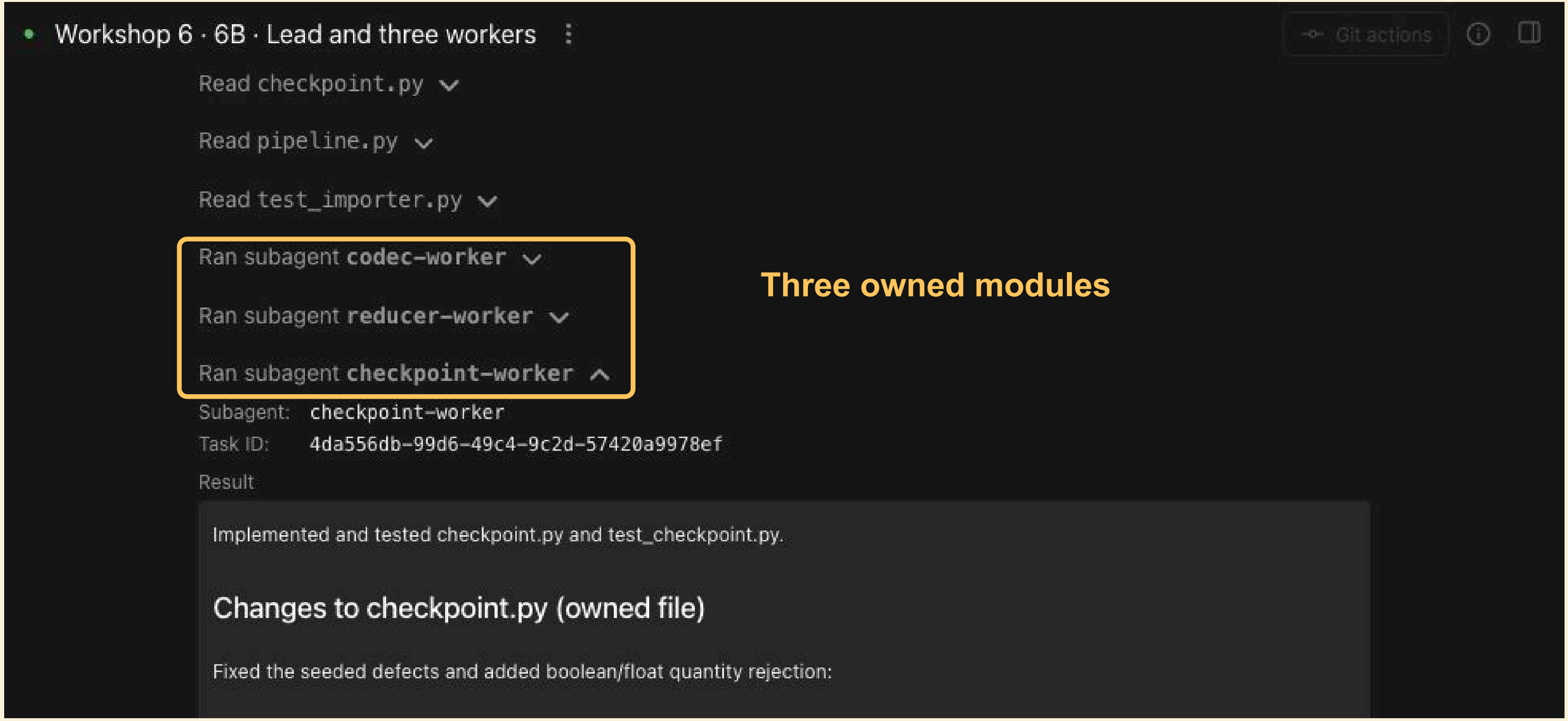


Open all five recorded conversations

One agent reached correct code in 80.5 seconds



Three workers repaired separate modules in parallel



Compare single to multi-agent - 🤑

Recorded setup	First correct code	Total recorded time	Total tokens	SDK cost
Single agent	**80.5s**	244s	647,914	$0.0551
Lead + 3 workers	**115.1s**	369s*	1,008,457	$0.1036

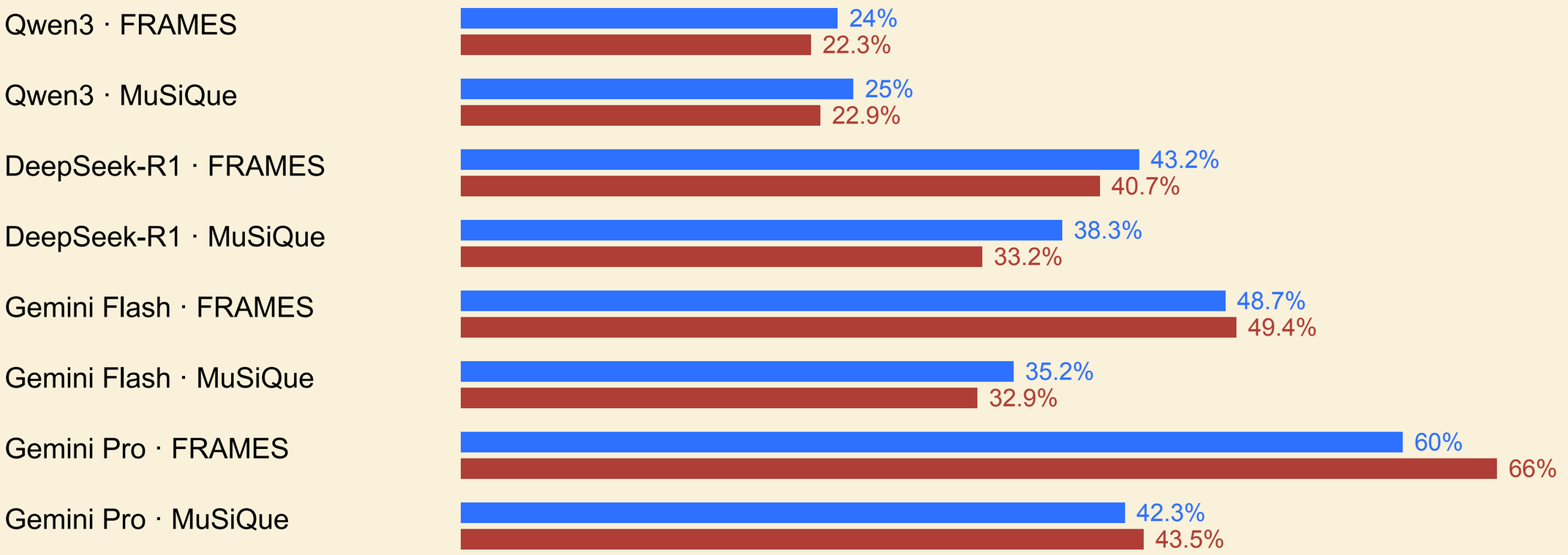
Both pass 33/33 independent checks

Team total: 56% more tokens · 88% more cost

*Lead hit the iteration cap without a final integration report.

At the same requested budget, multi-agent did not always win

500 requested thinking tokens · Best single-agent variant · Best multi-agent variant



Accuracy (%) · zero-based bars · higher is better

Point estimates; close gaps can overlap statistically. Gemini did not always use the requested budget.

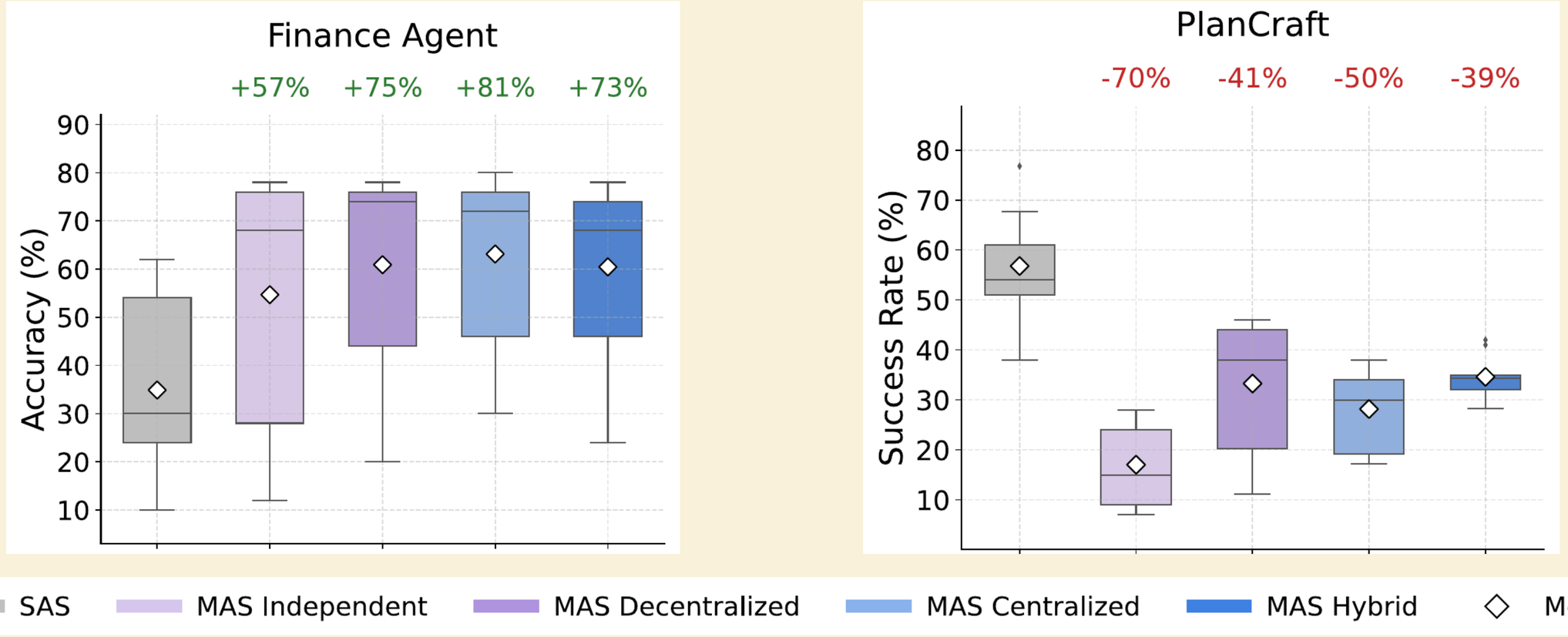
Tran & Kiela · Replotted from Table 1, 500-token columns · Best of 2 single-agent / 5 multi-agent variants



@rajistics

The same team architecture can help one task and hurt another

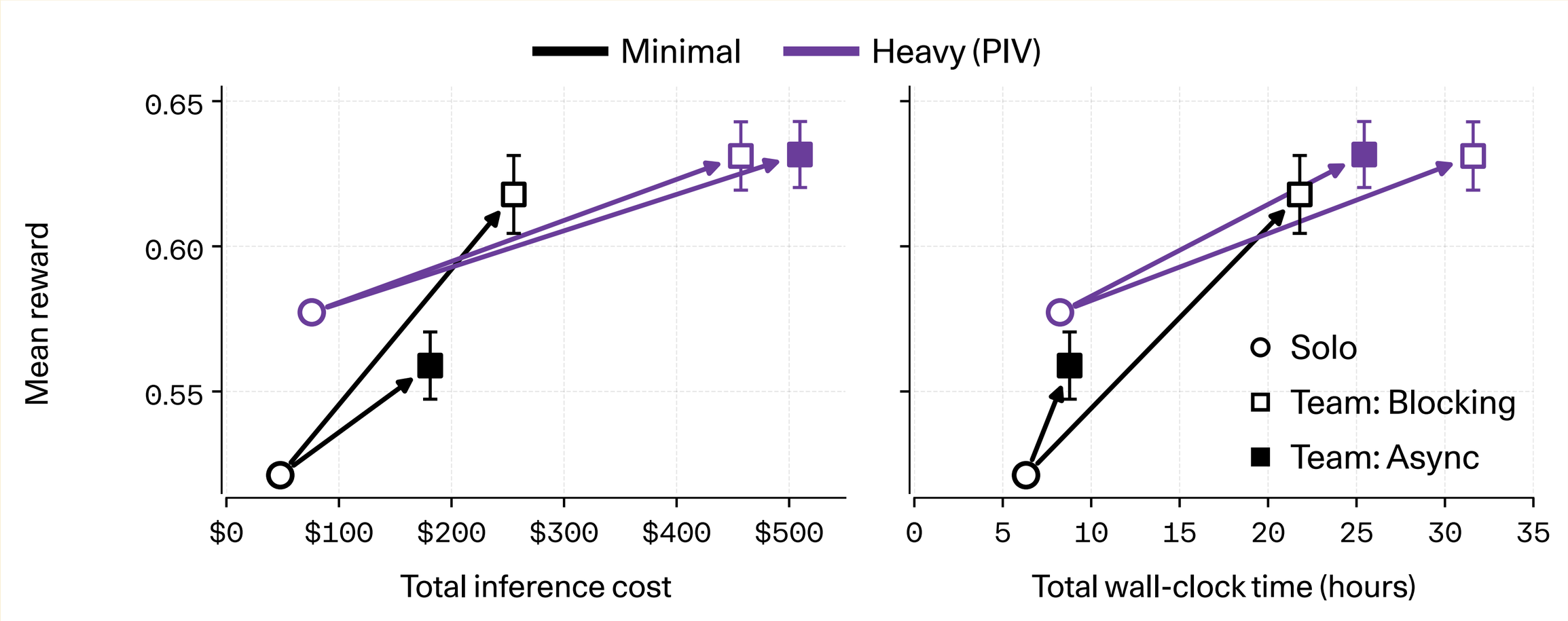
Kim et al.: 260 configurations · 6 benchmarks · 5 architectures · 3 model families



Kim et al. · Towards a Science of Scaling Agent Systems · Figure 2, selected panels · v3, April 2026

@rajistics

Coding teams improved results at 3–7× the cost



Cohere Labs · Samway et al. · Scaling Inference-Time Compute with Teams of Coding Agents · Figure 2
October 2026 preprint · Homogeneous Sol · ProgramBench: 200 tasks, 3 runs · Paired 95% intervals

Agent teams cost more, with one significant score gain

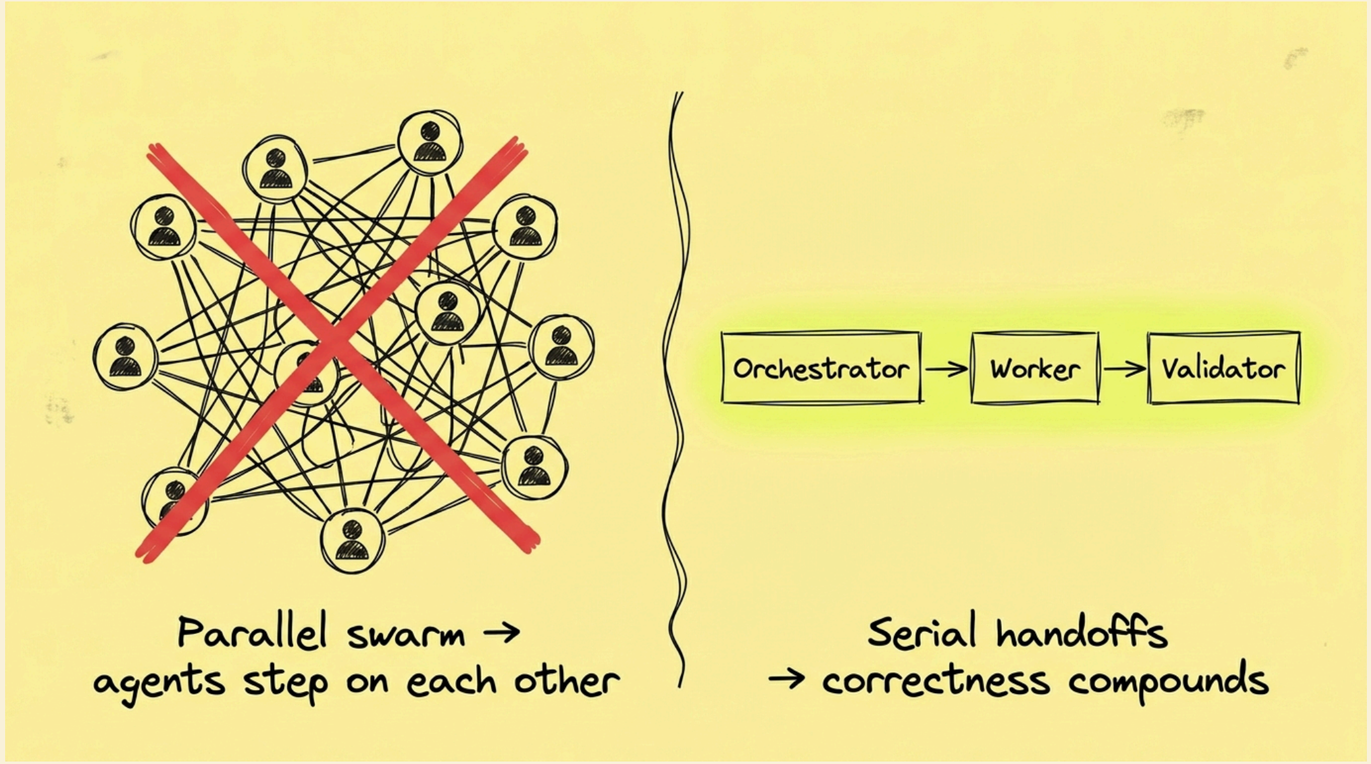
Vibe Code Bench: average share of UI tests passed across 50 web apps

Model and reasoning effort	Single agent	Team	Score change	Team cost
GPT-6 Sol, medium	77.6%	84.9%	**+7.3 pp***	2.5×
GPT-6 Sol, max	89.0%	90.4%	+1.4 pp	1.8×
Claude Opus 5.5, medium	91.5%	91.2%	−0.3 pp	2.2×
Claude Opus 5.5, max	89.8%	93.2%	+3.4 pp	5.1×

Vals AI, Do Agent Teams Pay Off?, October 9, 2026



Factory serializes code changes and parallelizes investigation.



Luke Alvoeiro, The Multi-Agent Architecture That Actually Ships

Was it the team, or the model?



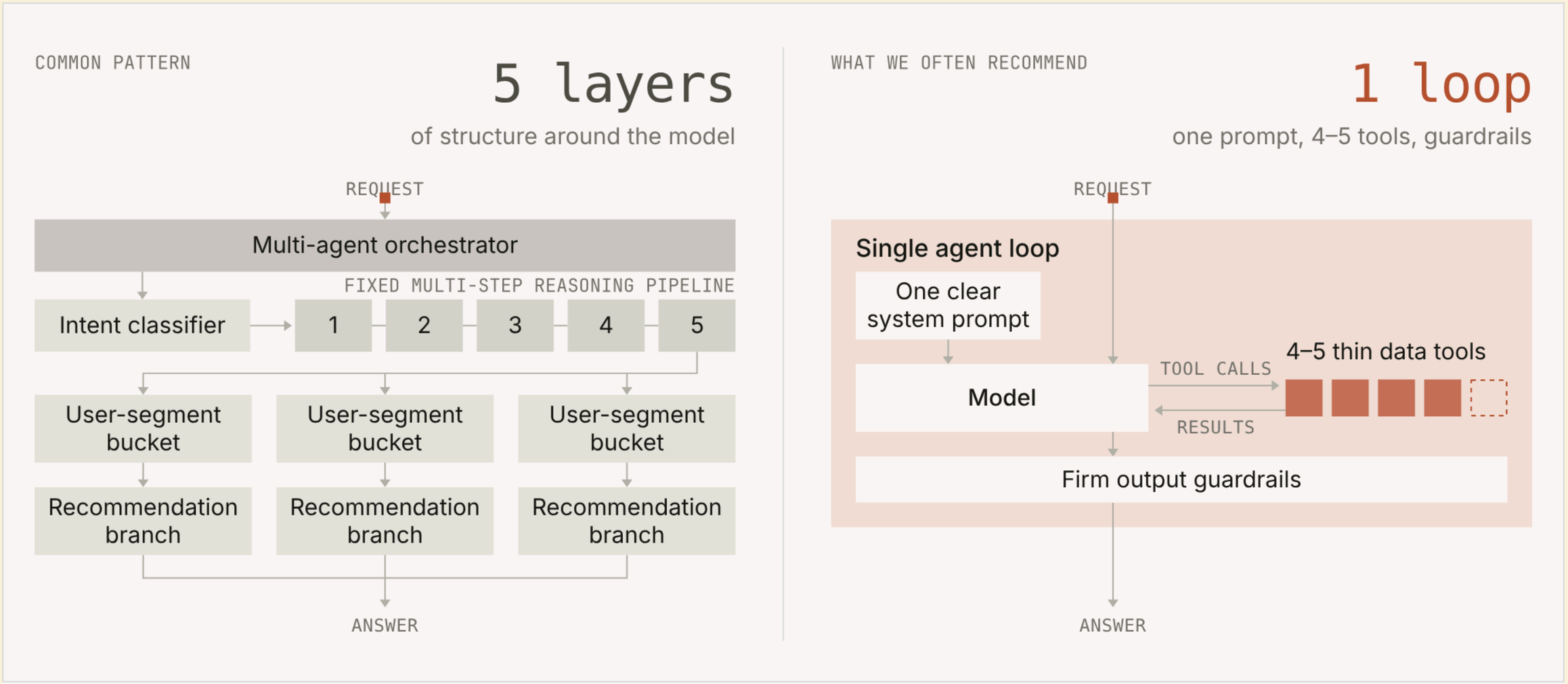
“I wouldn’t even attribute 10% of the credit to multi-agent.”

Noam Brown, OpenAI · on the Navier–Stokes result

Noam Brown, Dwarkesh Podcast · September 17, 2026. Interview assessment, not an ablation result.

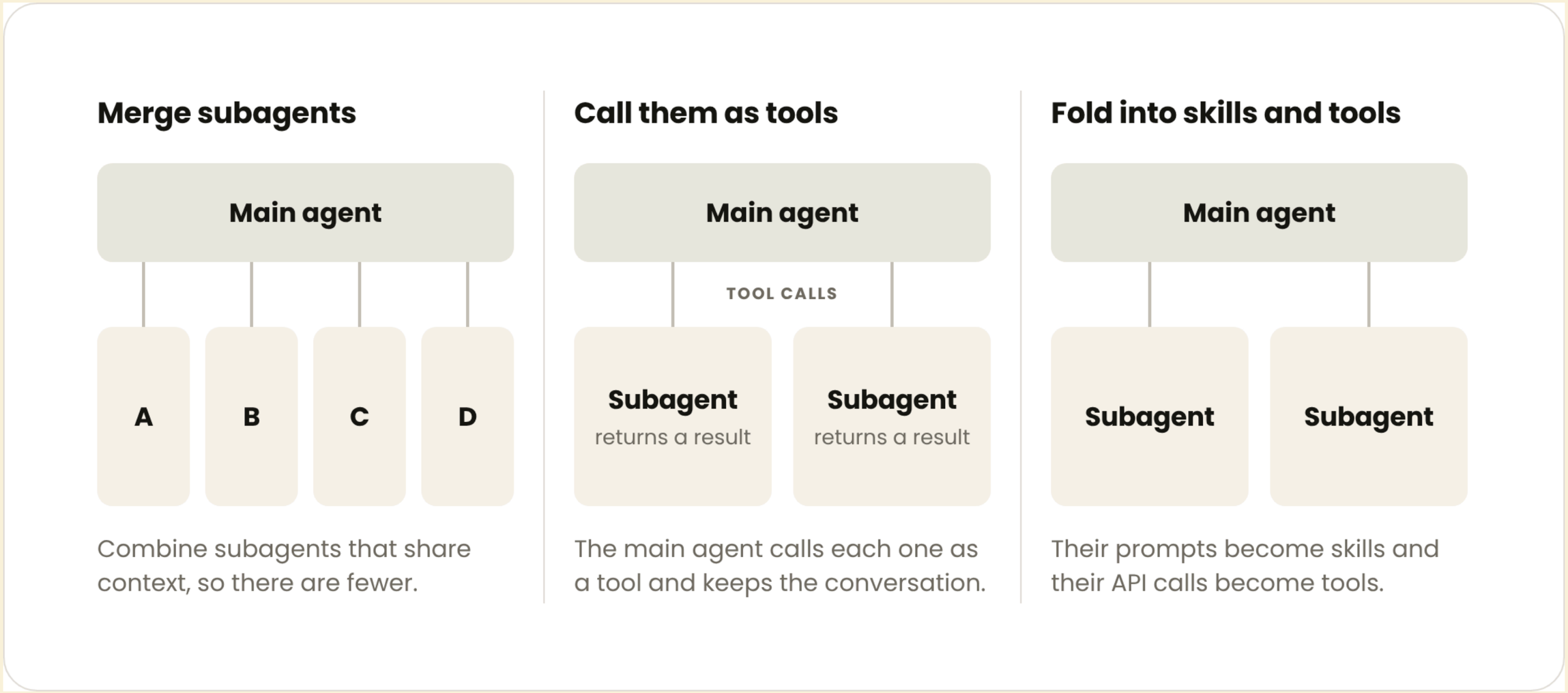
Shared context? Test a simpler agent loop

Keep subagents for independent parallel work, verification, or context isolation.



Original animation: Diane Sarkis / Anthropic · October 22, 2026 · Article ↗

Three ways to simplify multi-agent boundaries

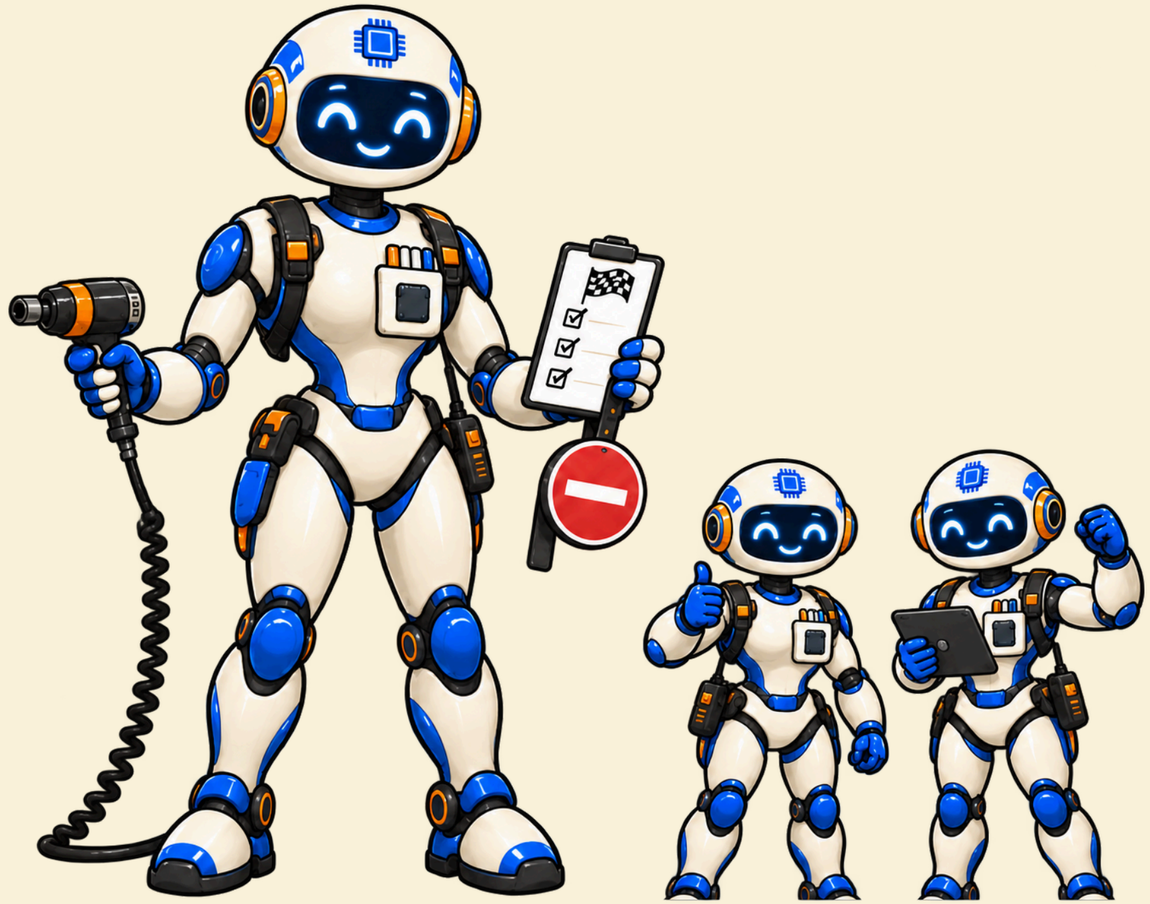


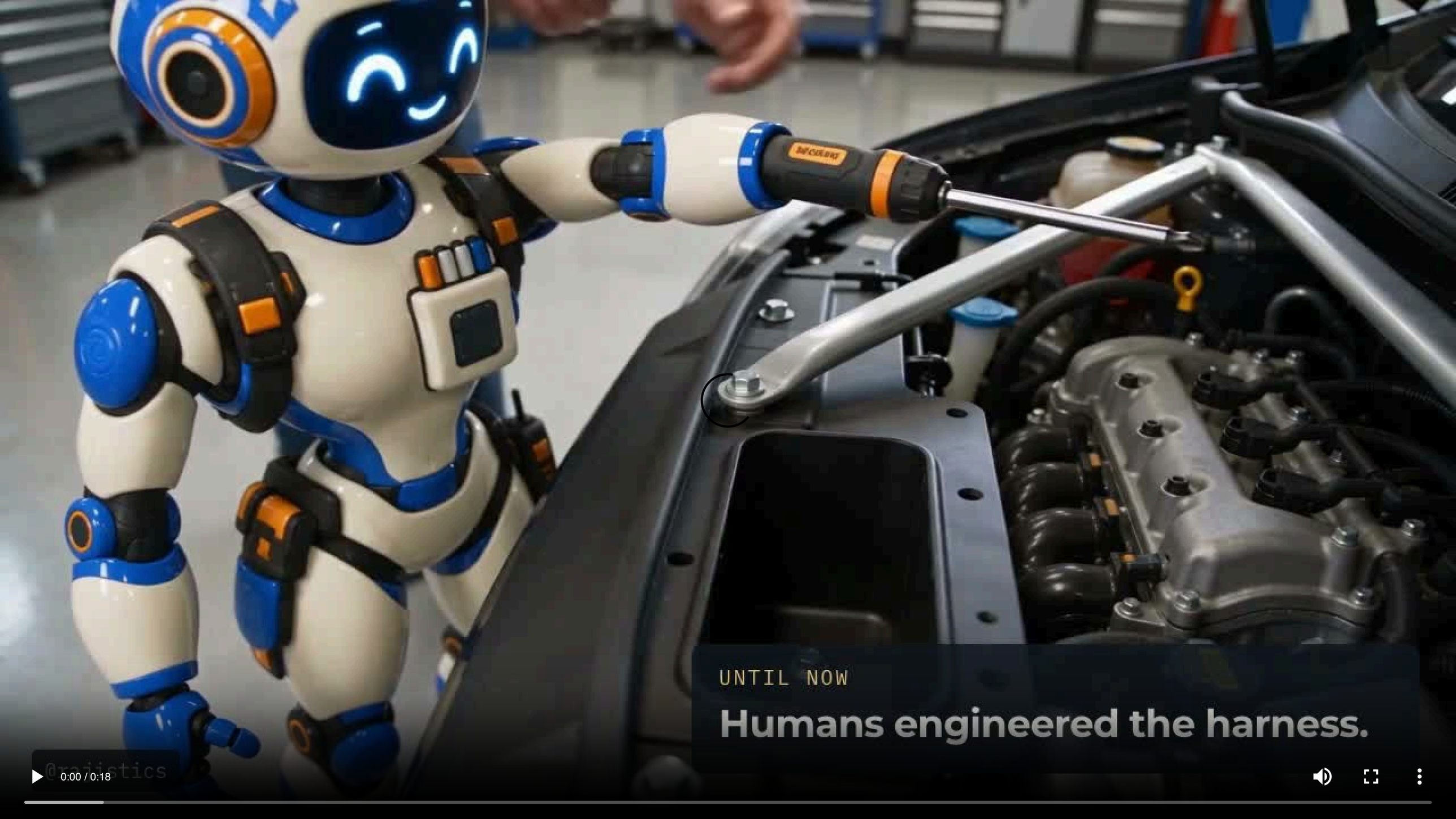
Original animation: Diane Sarkis / Anthropic · October 22, 2026 · Article ↗

SECTION

VII

Putting it together




UNTIL NOW
Humans engineered the harness.
0:00 / 0:18

Models learn behaviors. Harnesses enforce conditions



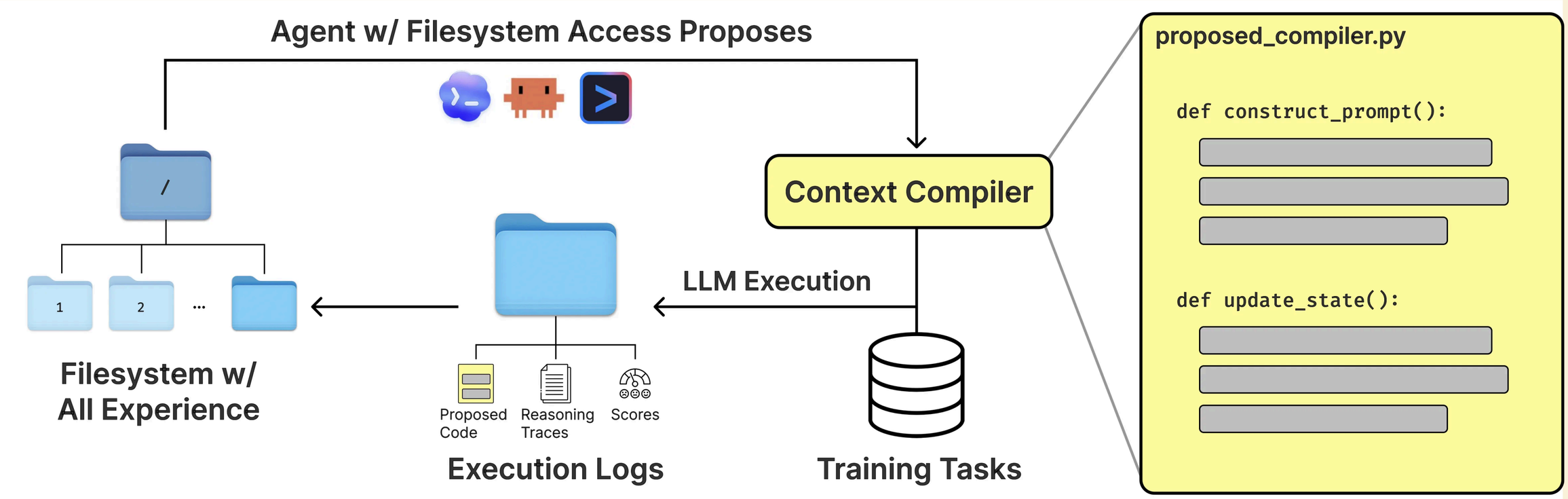
As the model improves, keep enforcing:

Evidence before “done”

Authorized actions

State and execution limits

Meta-Harness generates and improves harness code

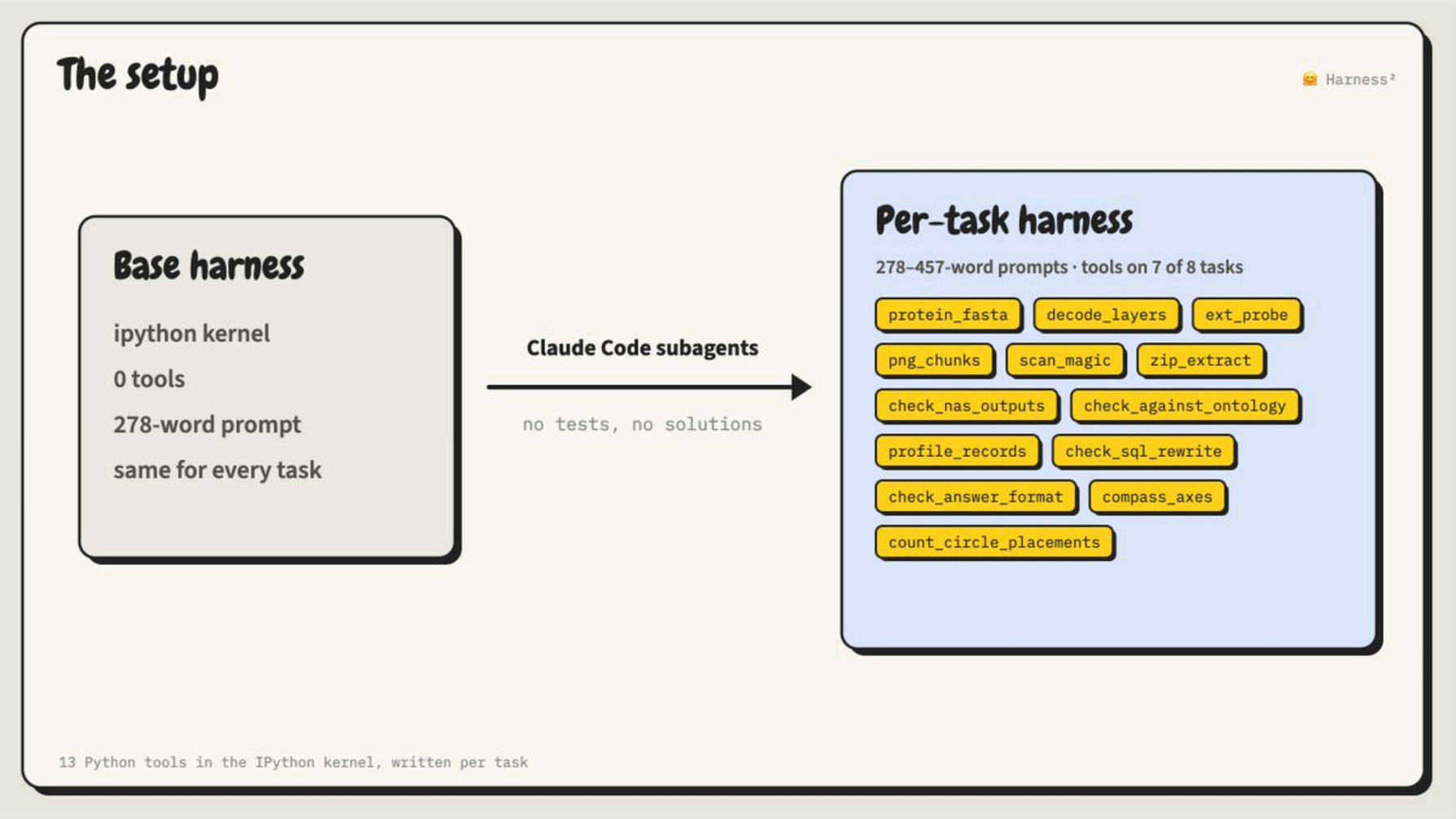


TerminalBench-2 pass rate

74.7% → 76.4%

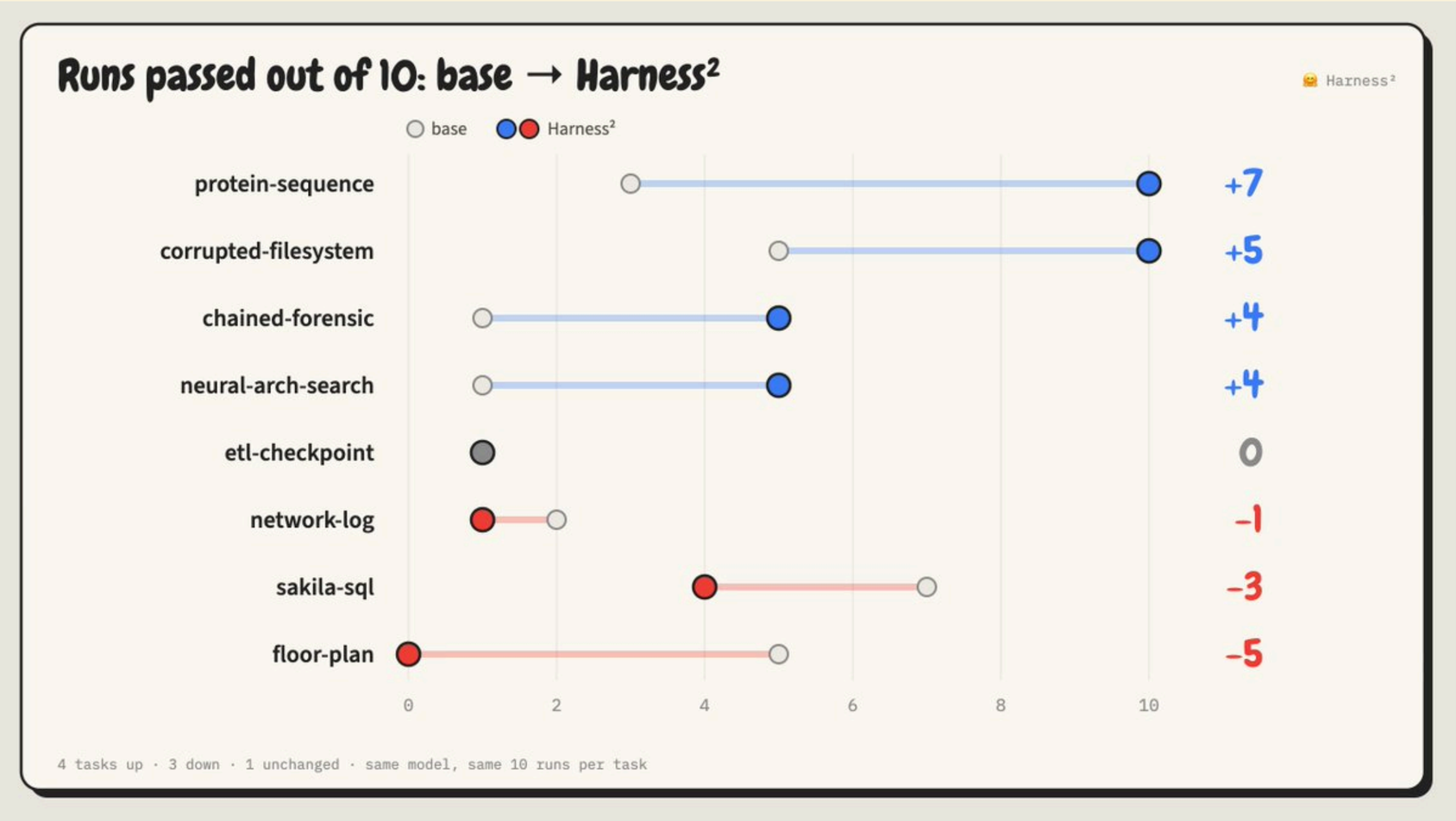
Lee et al. · Meta-Harness · Original method figure

The agent can redesign its harness for the task



Younes Belkada · October 23, 2026 · Original setup figure · Author-reported experiment

A tailored harness helps some tasks and hurts others



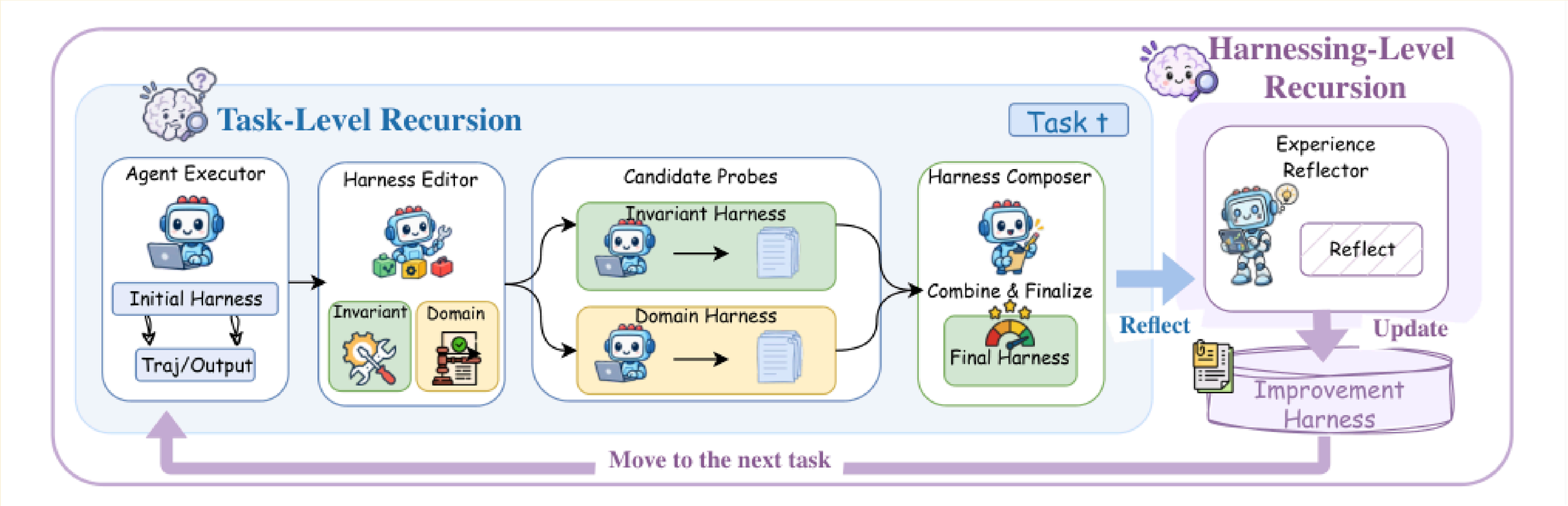
Mean pass rate

31%

→ 45%

Younes Belkada · Original results figure · 8 tasks, 10 runs per task and harness · Author-reported

Harness²: improve the harness, retain the lessons



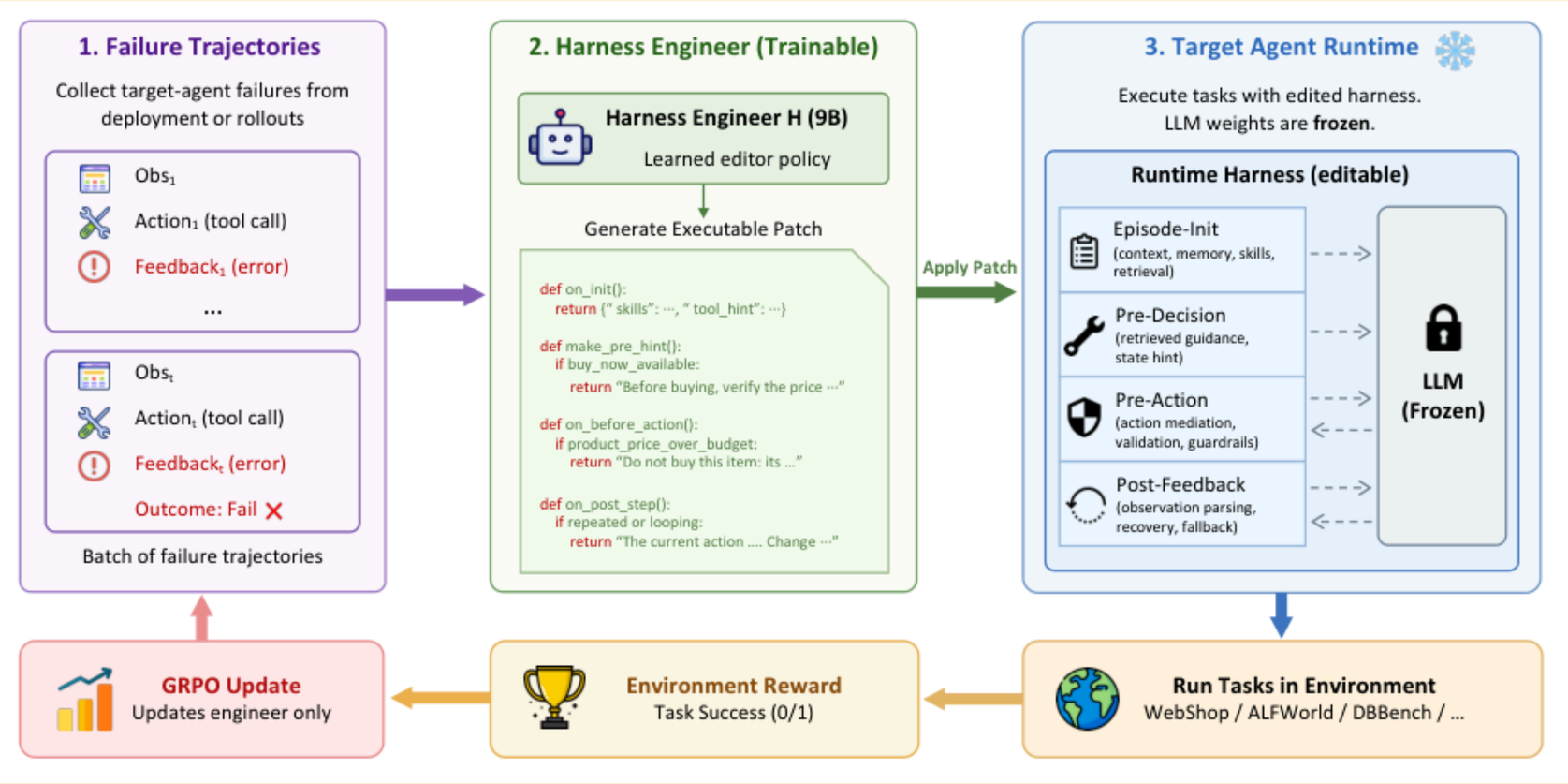
WorkBuddy-Bench average score

64.2 → 75.7

Xu et al. · Harness² · October 2026 preprint · Figure 3, Table 1

@rajistics

Harness-R1 learns to repair the harness from failures



Average task success

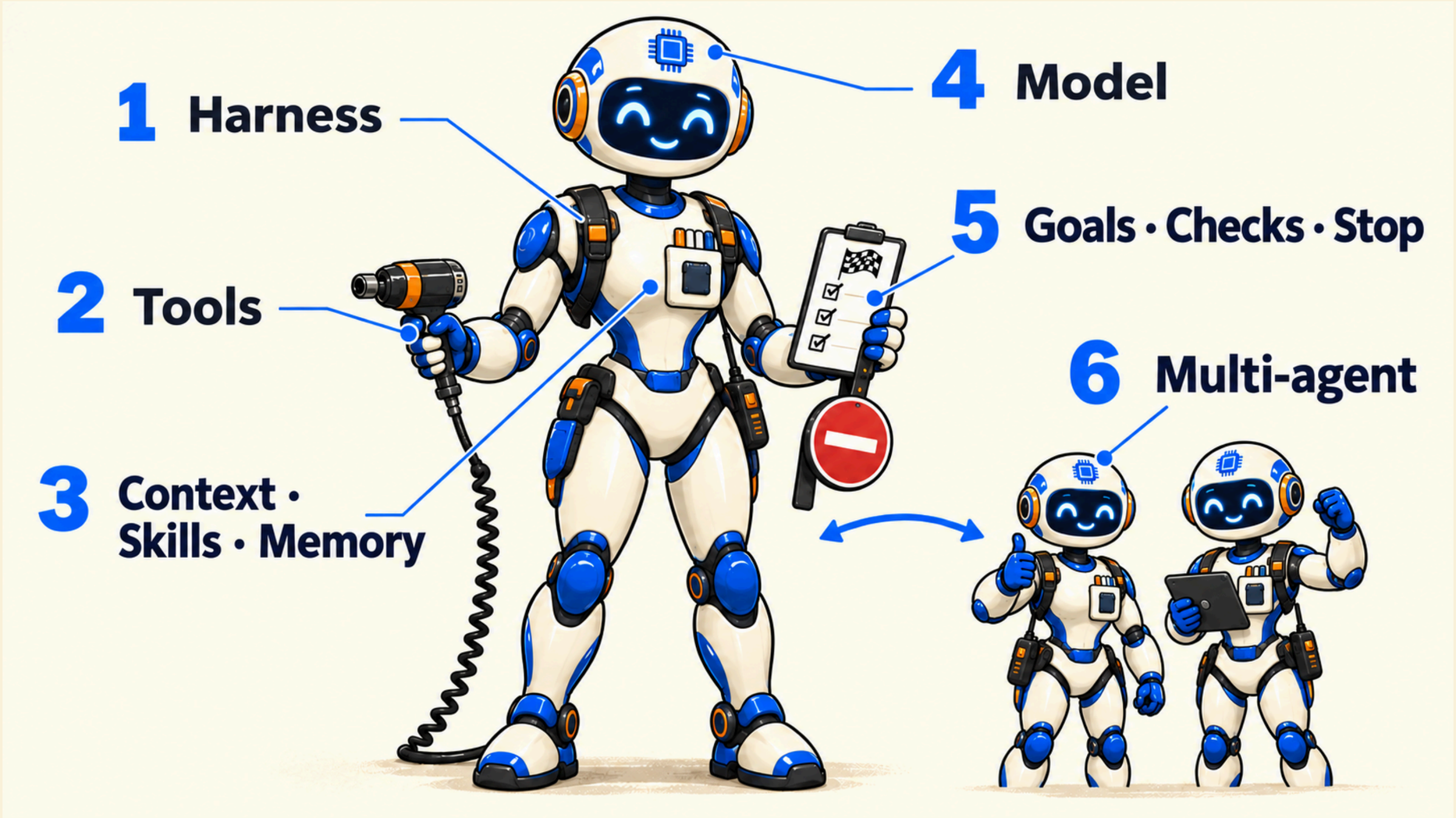
44.3%

→ 53.6%

Shao et al. · Harness-R1 · August 2026 preprint · Original Figure 2 · Fixed task-model weights

@rajistics

Whew! What we covered



What each harness decision can change

Harness	Execution behavior, reliability and overhead
Tools	Available actions and the feedback after each one
Context + memory	Relevant evidence now and continuity across tasks
Model	Capability, latency and inference cost
Goals + checks	What counts as success and when work stops
Team	Parallel capacity, handoffs and integration cost

Return to the six workshop labs

1 · Compare harnesses

2 · Design the tools

3 · Use and recheck memory

4 · Choose the model

5 · Verify the outcome

6 · Compare one agent with a team

Inspect in Canvas
or run the exercises

Canvas filter: lecture=odsc2026

is.gd/odsc2026_harness

Keep learning: CMU’s AI Agents course

CARNEGIE MELLON UNIVERSITY · 11-768 · FALL 2026

AI Agents

Graham Neubig & Daniel Fried

Tools, context, memory, planning
Coding agents, training, and safety



cmu-agents.com ↗

Lecture videos, slides, readings, and assignments