

Harness Engineering

or

How to Keep Agents On Track

About me

- Google Dev Expert
- Co-host @ Fragmented-AI
- Platform Engineering @ SumUp



@iurysza
iurysouza.dev

Non deterministic Coding

“They’ve done studies, you know?
60% of the time, it works every time”

- Brian Fantana (Anchorman)



Non deterministic Coding

- Agents are writing more and more code
- They write solid code, until they don't
- New bottlenecks
 - Keeping code quality
 - Steering
 - Containing blast radius

Non deterministic Coding

- Platform Teams attempts to keep quality standards for developers
- Harness Engineering does that for agents

Non deterministic Coding

- Prompt Engineering
- Context Engineering
- Harness Engineering

Non deterministic Coding



THIS TALK IS ABOUT LOOPS

What is the work we do?

INNER LOOP

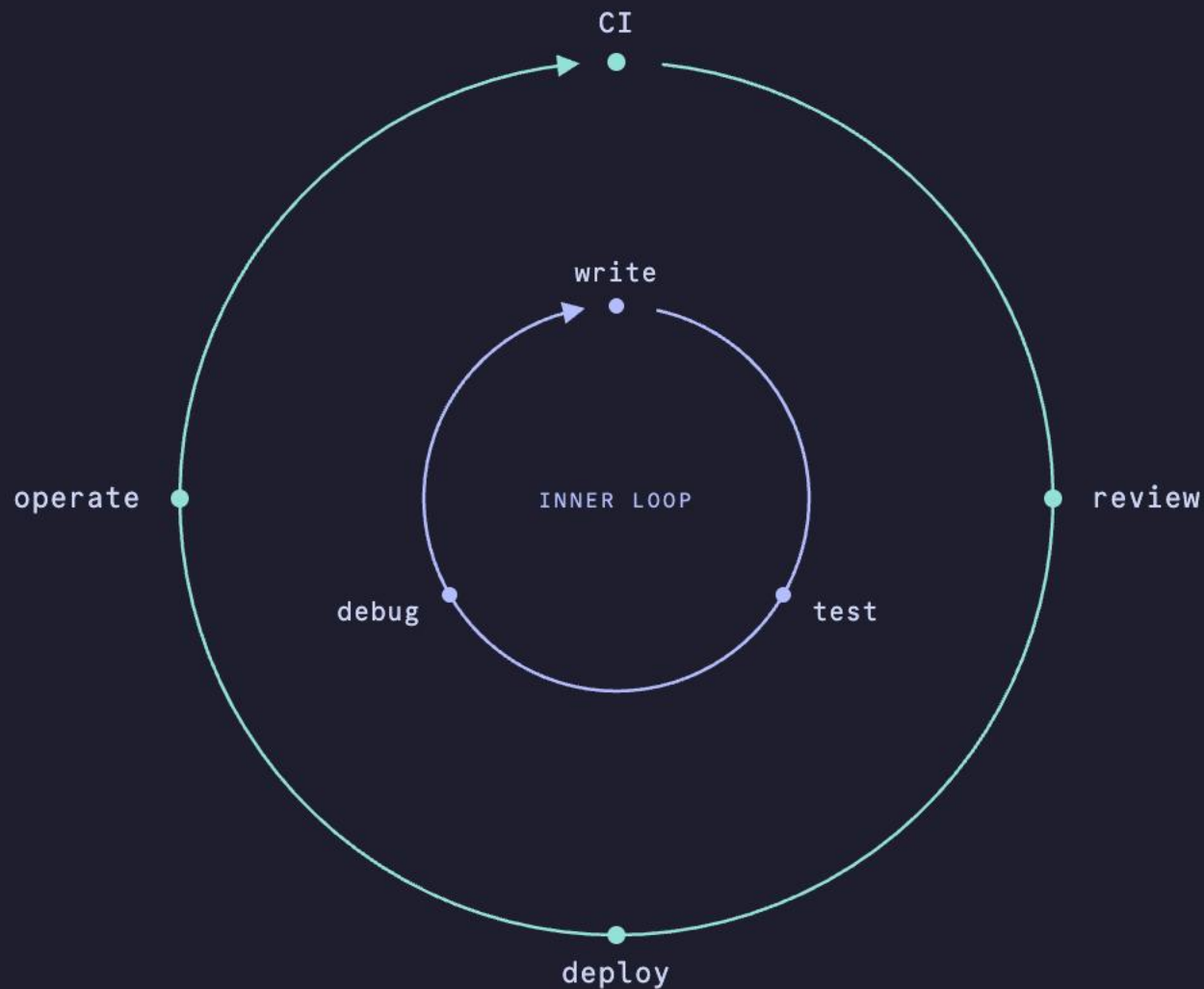
Your own cycle.

Write, test, debug, repeat.

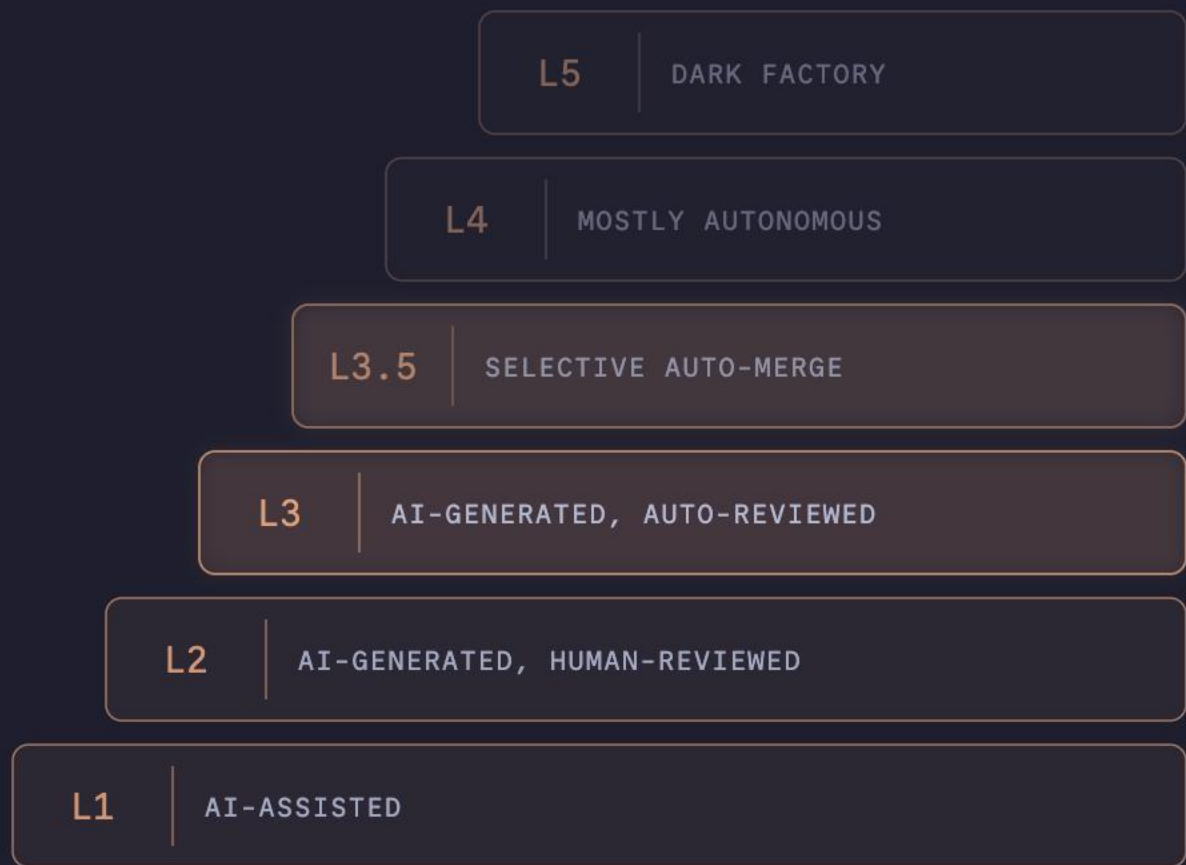
OUTER LOOP

The team's delivery cycle.

CI, review, deploy, operate.



How much of the loop do you hand over?



L3.5: AI-GENERATED, AUTO-REVIEWED

A reviewer agent can check the diff.
You skim the summary.

```
review      header typo fixed · low risk
ci          42 tests pass
you         read the summary · merge
```

WHO DOES WHICH JOB

PICKS WORK

YOU

WRITES

AGENT

REVIEWS

AGENT

MERGES

YOURISK

How we got Here

The Anthropic Moment

The Anthropic Moment

Feb 2025: Claude Code

- Claude Code preview: Feb 25
- Claude 3.7 Sonnet launch
- TUI / CLI based agent

The Anthropropic Moment

Nov 2025: Opus

- Opus 4.5
- Step change in quality
- Model can prompt itself
- Improved planning mode

The Anthropic Moment

Holidays: Claude Code + Opus

- The IDE starts losing ground
- The harness was the main improvement
- Competition shows up
 - Codex • Opencode • Amp • Pi
 - Gemini CLI • Copilot CLI • Cursor CLI • Droid • Crush
 - Goose • Kiro • Qwen Code • Kimi CLI • Cline CLI
- It's not magic
- The harness matters more than the model

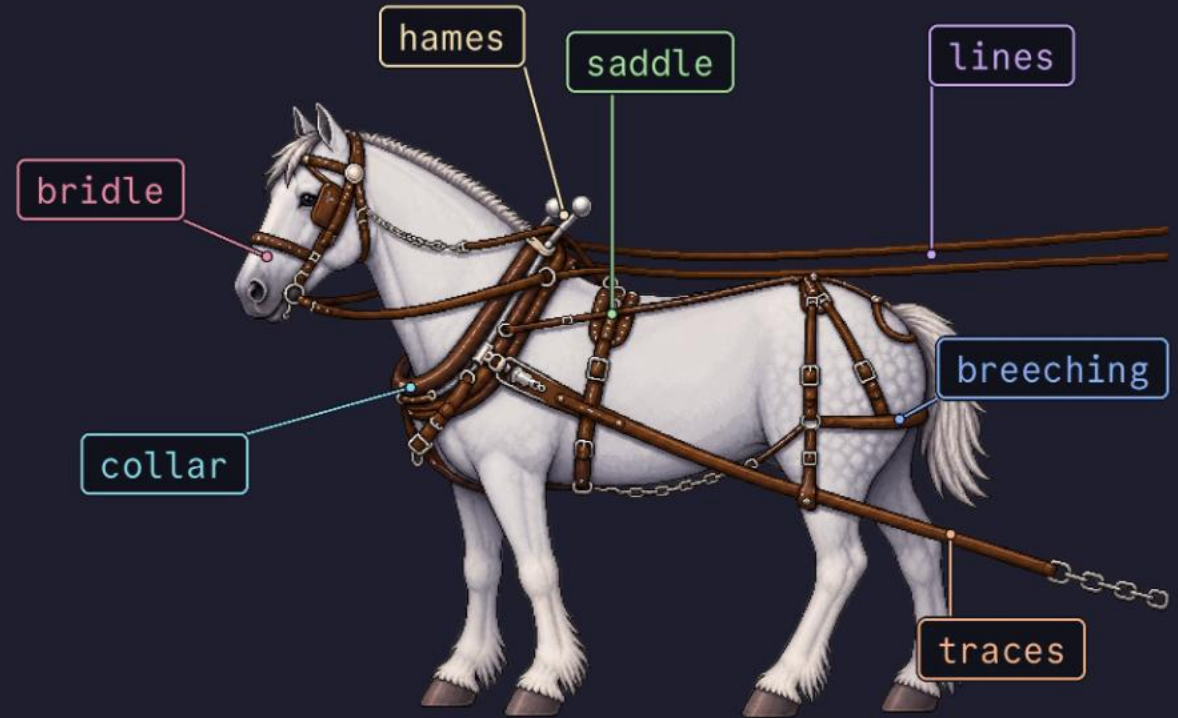
What is a harness anyway?

It's all about control

What is a harness

(...) a set of straps and fittings used to control an animal (...)

- Control
- Direction
- Constraints



A horse Harness

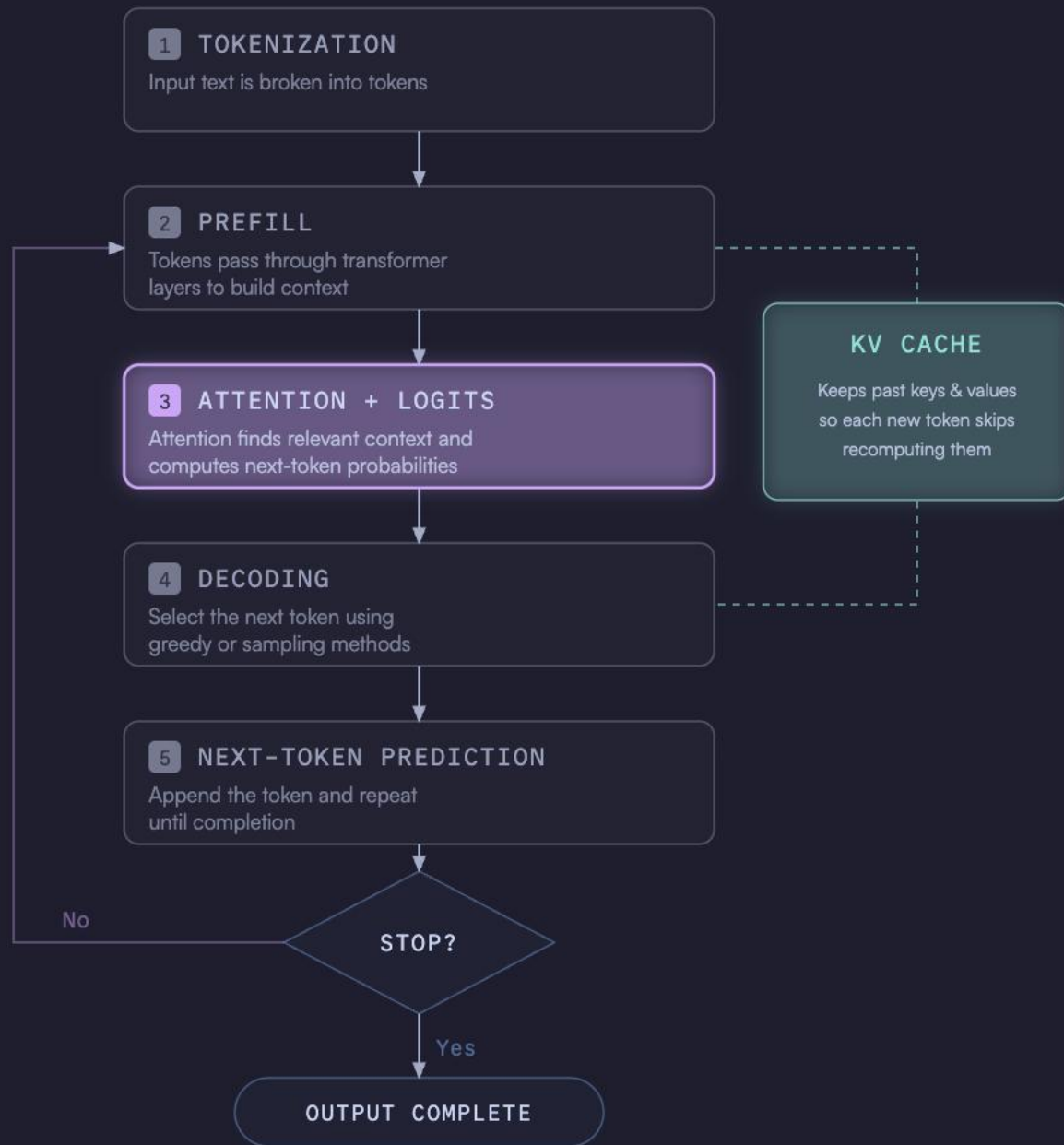
A model alone can't do anything

Text in, text out

- It can't read your files
- It can't run your tests
- It can't call an API
- To act in the world, it needs tools

Stacking Loops

What agents are made of



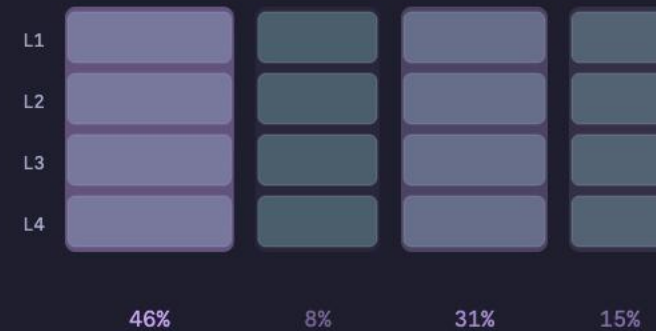
INNER LOOP · NEXT TOKEN GENERATION

ITERATION 1

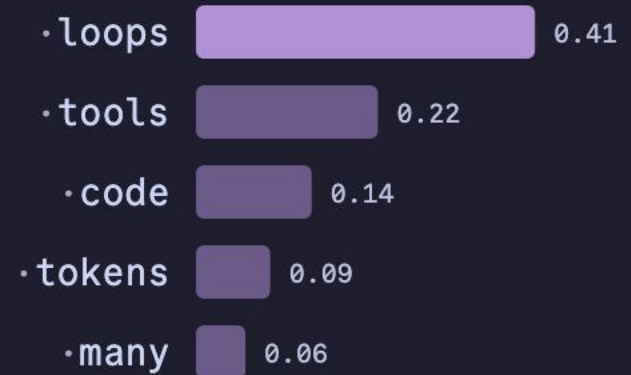
TOKENS



TRANSFORMER LAYERS → KV CACHE



NEXT-TOKEN PROBABILITIES



ReAct

Reasoning + Acting

- One pattern emerged: think, then act
- Reason → act → observe → repeat
- The harness runs that loop



USER GOAL

Why is the login test failing?

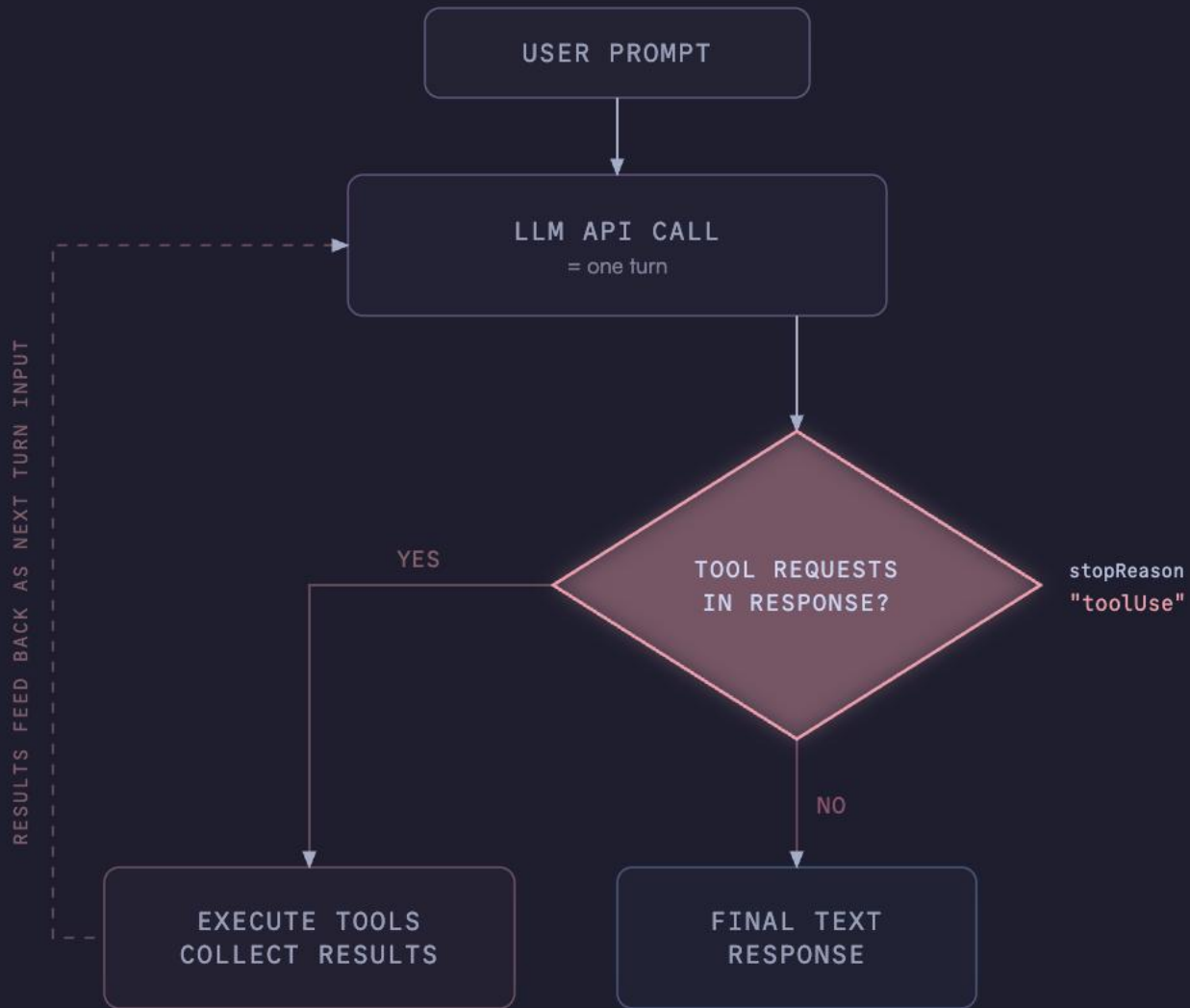
THINK → RUN → READ OUTPUT

- | | | |
|----|---------|---|
| 01 | REASON | Start with the test output. |
| 01 | ACT | <code>\$ npm test -- login</code> |
| 01 | OBSERVE | FAIL login: expected 200, got 401 |
| 01 | REASON | 401 means auth failed. Is the header right? |

Stacking Loops

Why this works?

- Non deterministic -> deterministic
- Code is verifiable
- Feedback loop



OUTER LOOP • THE AGENTIC LOOP

TURN 1

```
agent.ts

async function simpleLoop(messages, model, tools) {
  while (true) {
    const response = await callModel(model, messages, tools);
    messages.push(response);
    if (response.stopReason !== "toolUse") {
      return messages;
    }
    for (const toolCall of response.toolCalls) {
      const result = await executeTool(toolCall);
      messages.push(result);
    }
  }
}
```

MESSAGES[]

length 2

user Rename the prop title to heading in Card

assistant toolUse → edit("src/components/card.tsx", ...)

Should I build a Harness?

Short answer: no

- You can customize it
- You can extend it
- You can build infrastructure for it

TWO HARNESSES

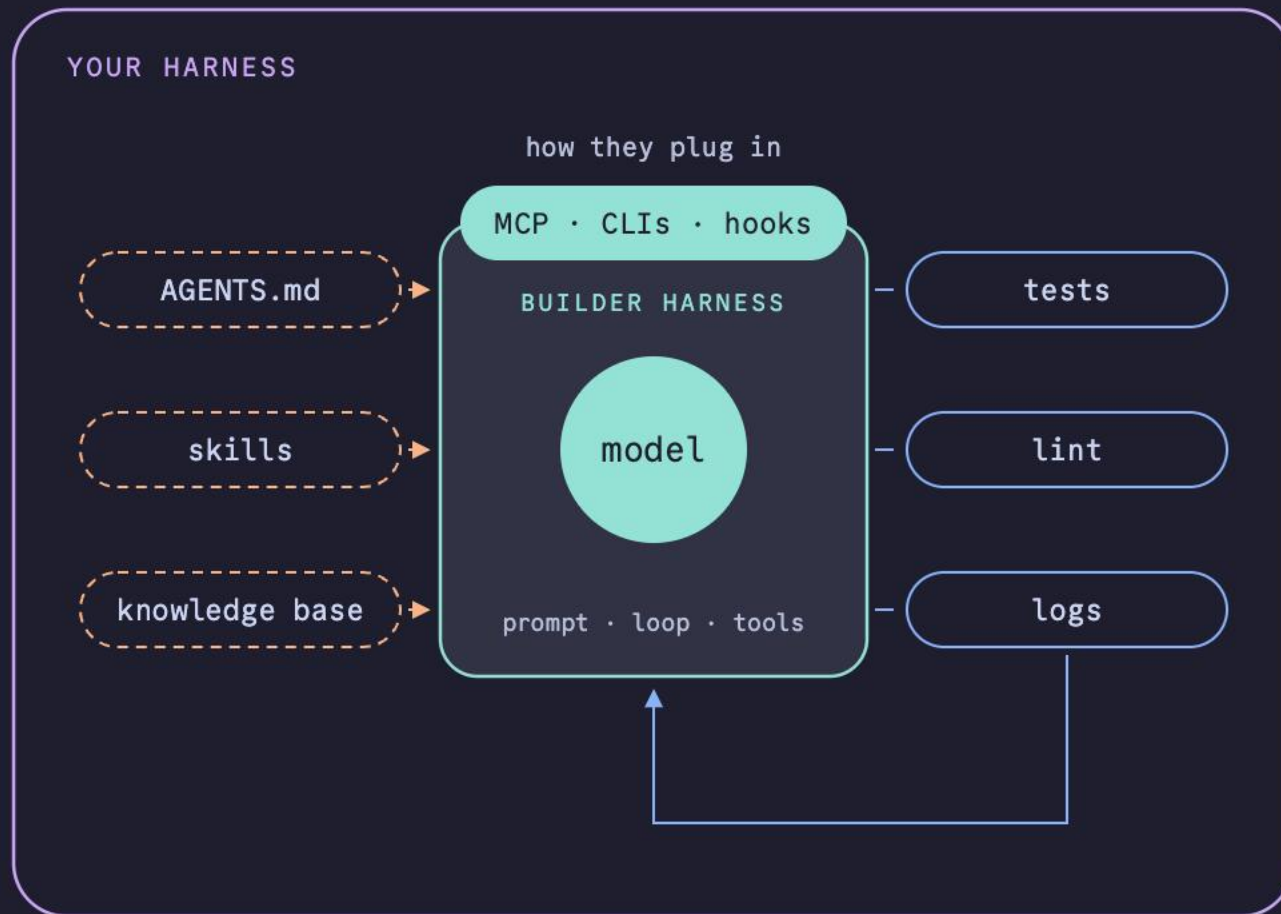
Whose harness is it?

BUILDER HARNESS

Claude Code, Codex, Pi.

YOUR HARNESS

Everything you add for your codebase.



Your harness

Four things you can add

- AGENTS.md
- A skill
- A hook
- An extension

Guides and sensors

Before it acts, after it acts

- Guides steer before the agent acts
- They make the first try right more often
- Sensors check after the agent acts
- They make a second try possible

Computational or inferential

Who does the checking?

- Computational: deterministic
- Fast, cheap, same answer every time
- Inferential: non-deterministic
- Can judge meaning, but slower and can be wrong

Before or after it acts? Code or a model?

GUIDES

feedforward · before it acts

COMPUTATIONAL

deterministic

CPU · ms

Make the right thing easy

Tools that shape the first attempt

CLIs

scripts

types

LSP

SENSORS

feedback · after it acts

Catch it, every time

Cheap enough to run on every change

tests

lint

type check

hooks

CI pipeline

INFERENTIAL

non-deterministic

GPU · s-min

Tell it what good looks like

Words the model reads before acting

AGENTS.md

skills

rules

reference docs

knowledge base

Judge what code can't

Semantic review, a model as judge

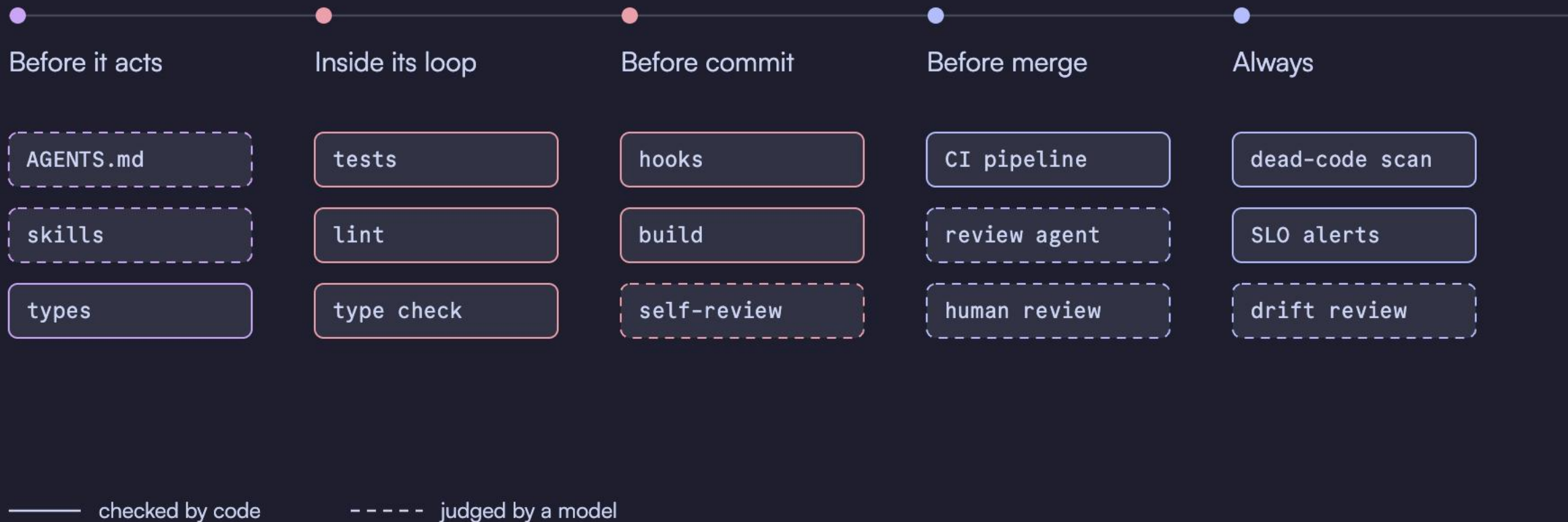
review agent

self-review

drift review

Keep quality left

The sooner a check runs, the cheaper the fix



Same mistake twice?

Fix the harness first, then the code

Closing the Loop

How to leverage the harness to make the agent smarter

THIS TALK IS ABOUT LOOPS

Closing the loop

INNER LOOP

Your own cycle.

Write, test, debug, repeat.

OUTER LOOP

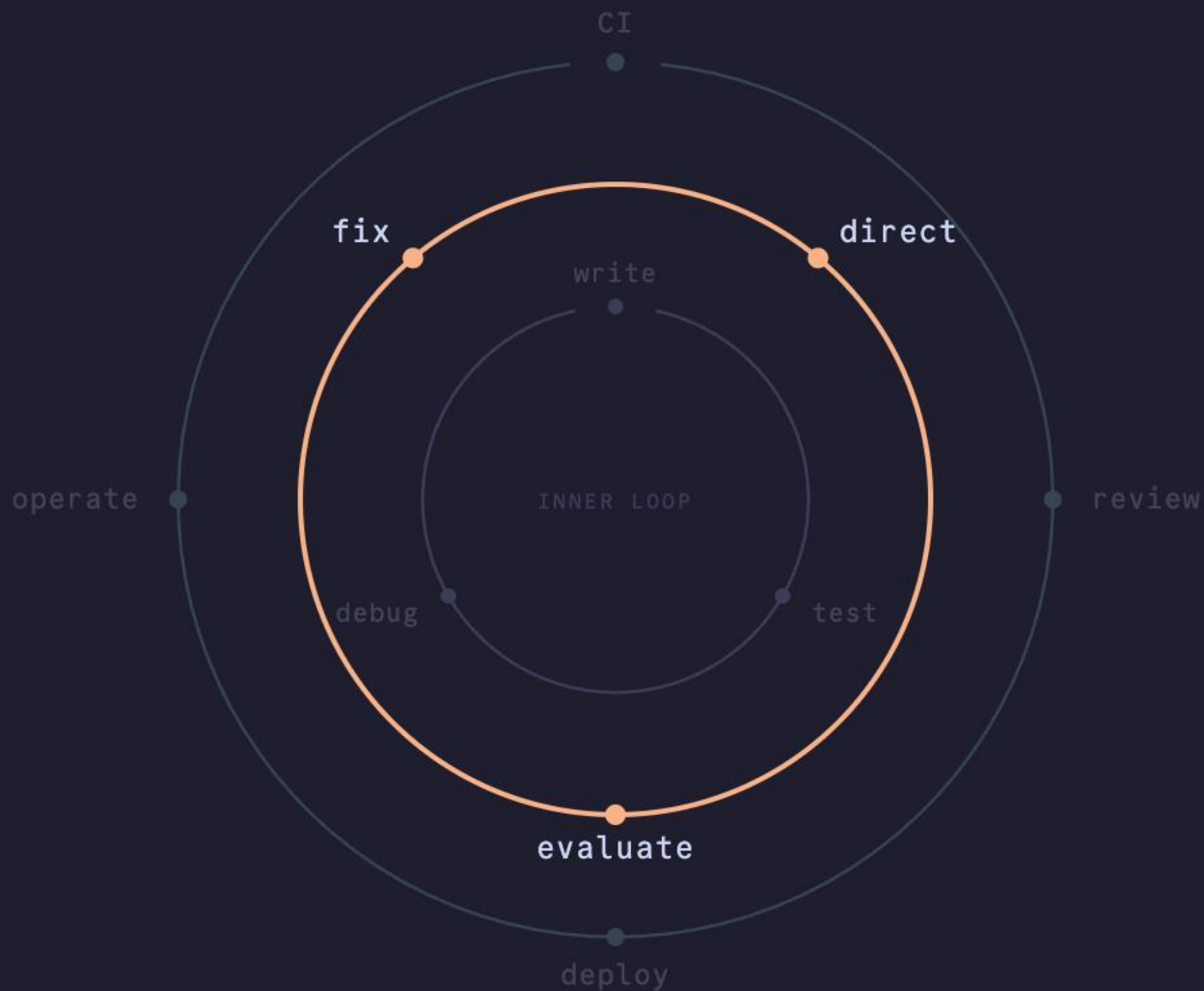
The team's delivery cycle.

CI, review, deploy, operate.

MIDDLE LOOP

Supervising agents.

Direct, evaluate, fix.



Closing the Loop

Give the agent senses

- Observability enables self correction
- Design agent friendly tools, CLIs/MCPs
- Favor them over raw text instructions

Closing the Loop

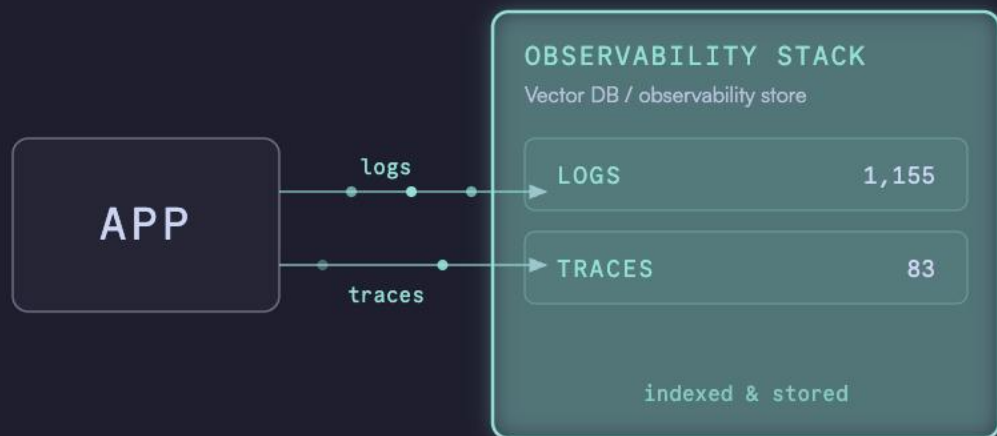
Give the agent senses

- Machine Readable Output
- Self describing schemas
- Safety rails against hallucinations
- Instrument the repository for agents

Closing the Loop

Give the agent senses

- Token Efficiency
- Reliability
- Reproducibility
- Auditable



■ 1. App emits logs & traces

■ 2. Stored & indexed

■ 3. Extra inspection data

■ 4. Agent queries all sources

Every app needs its own harness

Even OpenAI couldn't ship an empty box

- Teams add their own skills and plugins
- Experts bring the taste
- Docs live next to the code
- Your app is a domain-specific harness

Two camps

Why work on a harness at all?

- The bitter lesson: wait for a better model
- Neuro-symbolic: the model needs code around it

The bitter lesson

Wikipedia

“In the long run, general approaches that scale with available computational power tend to outperform ones based on domain-specific understanding.”

The bitter lesson

Rich Sutton, 2019

“Building in **how we think we think** does not work in the long run.”

Neuro-symbolic AI

Wikipedia

“Combines neural networks and symbolic AI ... to create more robust, more reliable, and more trustworthy AI.”

Neuro-symbolic

Gary Marcus, 2026

“Claude Code isn’t better because of scaling. It’s better because it is **neurosymbolic**.”

Bigger, not smaller

Thariq Shihpar, Anthropic

“The models get better and better. And so the harness needs to become **more and more complicated** to allow the model to do more things.”

The middle ground

Both are right, about different layers

- Scaffolding that teaches it to think: the next model eats it
- Checks that connect it to your world: they stay
- Don't teach the model to think. Give it ways to check.

Human input

Direct it, don't remove it

“A good harness should not necessarily aim to fully eliminate human input, but to direct it to **where our input is most important.**”

Platform Engineering

And the tragedy of the commons

Platform Engineering

The tragedy of the commons

An economic and ecological concept describing how **individuals**, acting strictly in their own **self-interest**, deplete or spoil a shared resource, ultimately **ruining it for everyone**

Platform Engineering

Enables Speed and Cohesion

- Company Scaling Implications
 - Shared standards
 - Tooling
 - Guardrails

Platform Engineering

Dedicated Agent Infra

- Fast moving environment
- Agent Evals and benchmarking
- Skills versioning and distribution
- Safely expose company systems for agents
- Research and Development

Conclusion

Experiment & Iterate

What we covered

The talk in one slide

- A model alone can't act. The harness gives it a loop.
- Guides before it acts, sensors after
- Same mistake twice? Fix the harness first.
- There is now a new middle loop
- Someone needs to own the commons

Conclusion

Experiment & Iterate

- Mastering Harness engineering
 - Enables shipping faster
 - Fewer regressions
 - Lower Organizational drift
- No one-size-fits-all
- Start experimenting and find out what works best for your team/org

Sources

Read these next

- Böckeler, "Harness engineering", martinfowler.com
- Yao et al., "ReAct", 2022
- Sutton, "The Bitter Lesson", 2019
- Marcus, "The biggest advance in AI since the LLM", 2026

Harness Engineering

or

How to Keep Agents On Track

Keep reading

My blog and newsletter

- More on harnesses, agents and loops
- iurysouza.dev
- Link and QR on the last slide

Thank you!



@iurysza
iurysouza.dev

Questions?



@iurysza
iurysouza.dev