CS224G

Building AI-Native Companies

And Evaluating Agents Without Losing Your Mind

Ryan Brandt

Vunda AI

Today's Agenda

1 **AI-Native Operations**
What it actually means

2 **Automated Consulting**
Spec-driven development

3 **Agent Evals**
179 failures analyzed

4 **The OpenClaw Phenomenon**
Where the industry is now

5 **heynoah.io** — What I'm building · Clawdbot for everyone else

PART 1

What Does "AI Native" Mean?

What the Fastest Are Doing

Andrej Karpathy
@karpathy · Dec 2025

"The profession is being dramatically refactored..."

Dario Amodei
Anthropic CEO · Mar 2025

"AI will be writing 90% of the code."

Kent Beck
Creator of TDD · Dec 2025

"AI collapses the search space. Tasks that took days take hours."

Boris Cherny
Claude Code Creator · Jan 2026

"80-90% of Claude Code is written using Claude Code."

Theo (t3.gg)
@theo · Dec 2025

"70-90% of code is now directly generated."

This isn't hype.
It's what top performers are doing *right now*.

What AI Native is NOT

- ✗ "We use ChatGPT for customer support"
- ✗ "We have an AI chatbot on our site"
- ✗ "Our engineers use Copilot"
- ✗ "We fine-tuned a model once"

These are **AI-assisted**. Not AI-native.

What AI Native IS

System design where AI is a
first-class execution layer

Not a tool you use. An **actor that does work**.

The Test

Can unstructured input become
executed work product
with zero human in the loop?

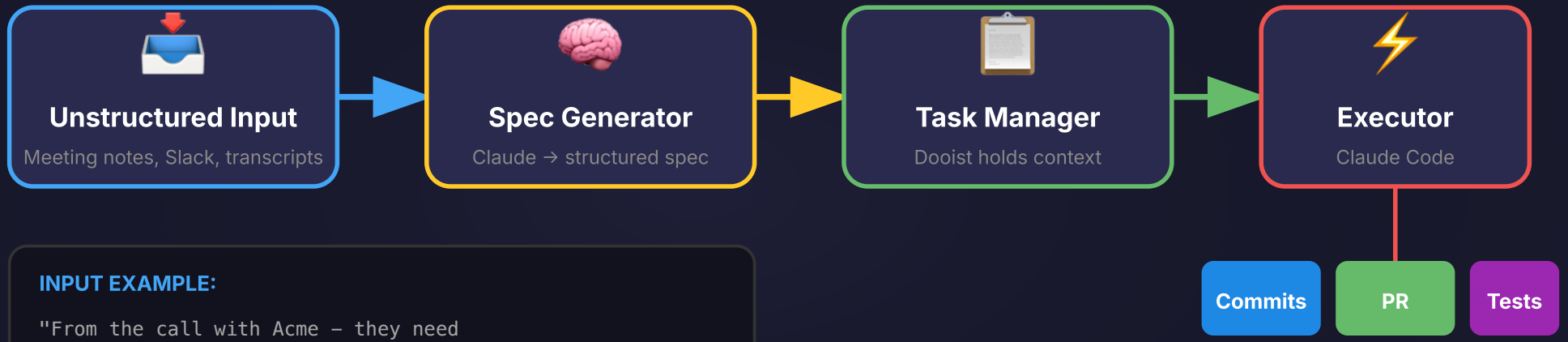
Meeting notes → Code → Commit → **PR**

PART 2

Automated Consulting Pipeline

Live demo time

The Pipeline



INPUT EXAMPLE:

```
"From the call with Acme - they need
login working by Friday. JWT is fine.
Sam mentioned bcrypt for passwords."
```

OUTPUT:

```
✓ src/auth/jwt.py created
✓ tests/test_auth.py passing
```

The Spec Format

```
Repo: ~/git/acme-api
Client: acme_corp
Intent: Add JWT-based user authentication to the REST API

## Done When
- [ ] POST /api/auth/login accepts email/password, returns JWT
- [ ] POST /api/auth/register creates user with hashed password
- [ ] Protected routes return 401 without valid token
- [ ] Token expiry is enforced (24h)
- [ ] All tests pass with `pytest tests/`

## Requirements
- JWT tokens with 24-hour expiry
- bcrypt for password hashing (cost factor 12)
- Return 401 for invalid credentials
- Email must be unique, validated format

## Key Files
Files to create:
- src/auth/jwt.py - Token creation/validation
- src/routes/auth.py - Login/register endpoints
- tests/test_auth.py - Auth tests

Files to modify:
- src/main.py - Register auth routes
- src/models/user.py - Add password_hash field
```

Context
Where & what

Done When
Verifiable checklist

Requirements
Constraints & decisions

Key Files
Explicit scope

Why This Works

✓

Checkable

Every "Done When" can be tested

Scoped

Key files listed explicitly

Contextual

Architecture pre-decided

Executable

Agent runs autonomously

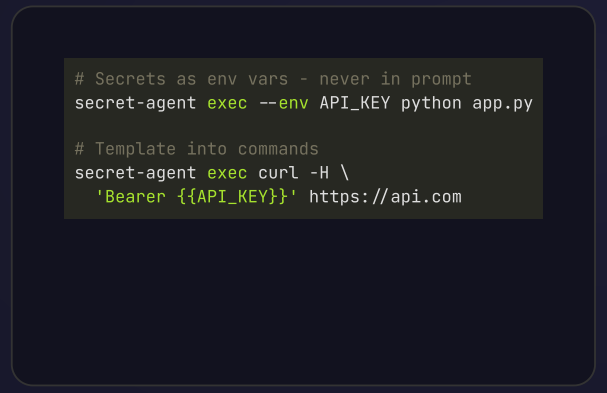
Demo: Transcript → PR

```
# From meeting notes to spec
python pipeline.py spec notes.txt

# Execute spec in target repo
python execute_spec.py \
    --spec spec.md \
    --repo ~/git/client-project \
    --pr

# Output:
# ✓ Spec validated
# ✓ 3 files created, 2 files modified
# ✓ Tests passing
# ✓ PR created: github.com/acme/api/pull/47
```

[Live demo of pipeline execution]

Keeping Secrets Safe: secret-agent

Open source CLI vault for AI agent workflows

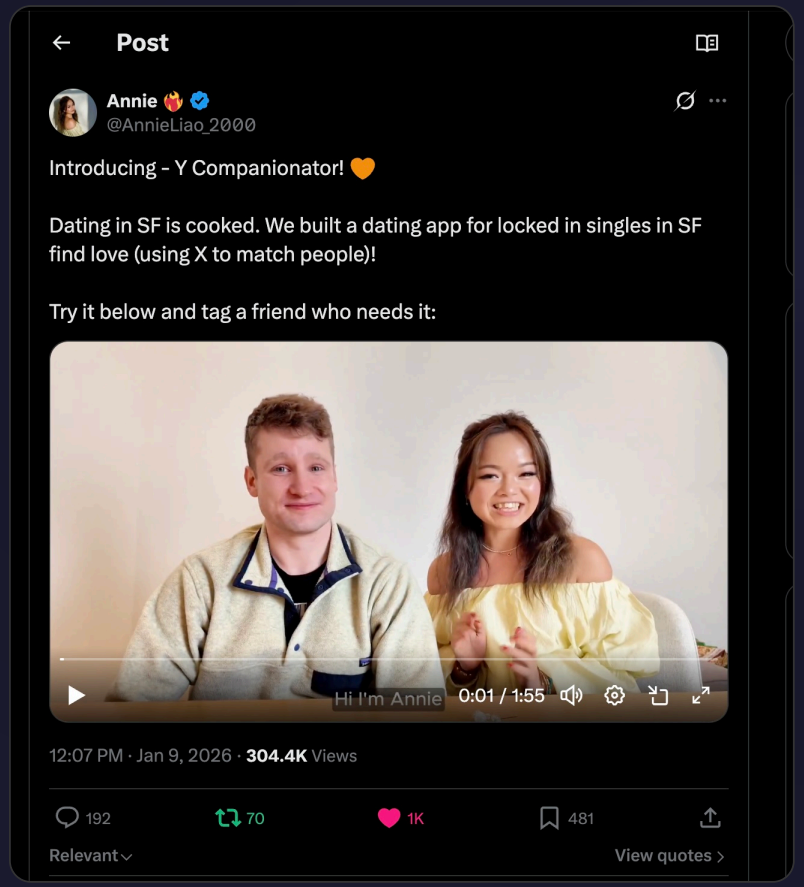


- Secrets never in context
- Traces safe to log
- Output auto-redacted

github.com/paperMoose/secret-agent

Real Example: YCompanionator

Conversation → Spec → Ship → Results



352 profiles

407 emails sent

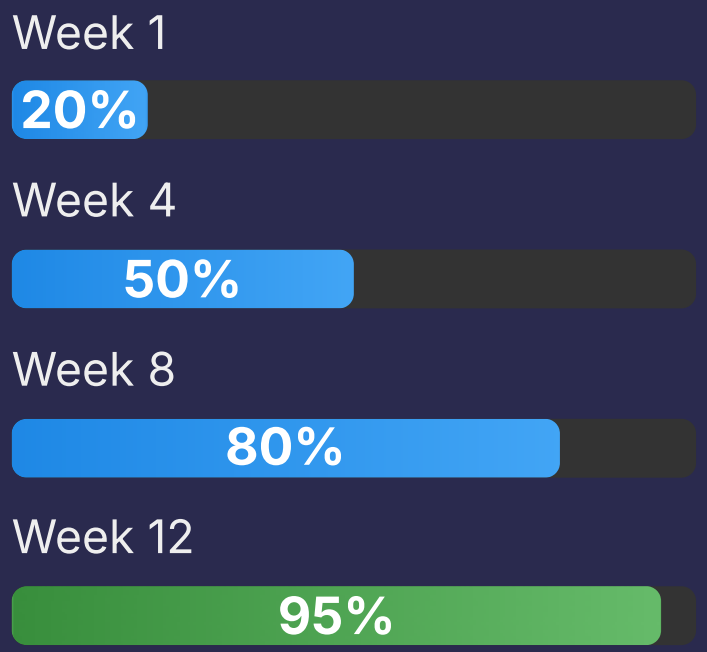
81% said YES

5 real couples 💕

The process:

Call with Annie → Spec → Ralph loop → **Shipped**

Context as Appreciating Asset



Every interaction adds to the context:

- Client preferences accumulate
- Past decisions inform future ones
- The more you use it, the better it gets

Your context file is your moat.

PART 3

Agent Evaluation

Prompt-driven first, error-driven second

Two Types of Failures

Spec Failures

Your prompt was incomplete or unclear

Example:

"Schedule a quick sync"

→ Model guesses 60 min (you meant 15)

✓ **Fixable with better prompts**

Generalization Failures

Model can't extend rules to new situations

Example:

Prompt covers 20 edge cases perfectly

→ Case #21 still breaks it

⚠ **Route to human or accept limitation**

PHASE 1

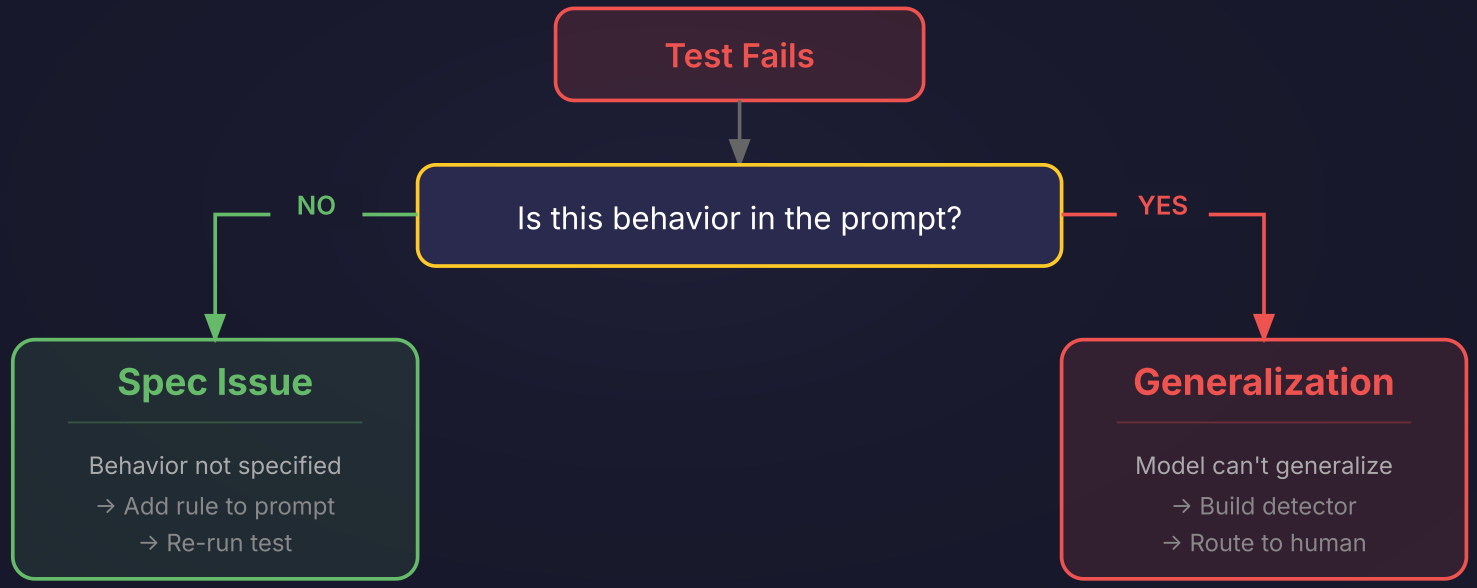
Prompt-Driven Evaluation

Your prompt is your test spec

1. Every rule in your prompt → a testable assertion
2. Generate test cases *from* the prompt itself
3. Run model → **failures surface issues**
4. Classify: **spec issue** or **generalization issue**?

Classifying Failures

The test tells you WHAT failed. You determine WHY.



Knowing When You've Hit the Wall

Prompt Overload

- Too many rules competing
- Fixing one breaks another
- Model gets confused
- Prompt is pages long

Fix: Simplify scope, split into multiple agents

Generalization Limit

- Prompt is clear and minimal
- Model still can't do it
- Edge cases keep appearing
- Requires reasoning model can't do

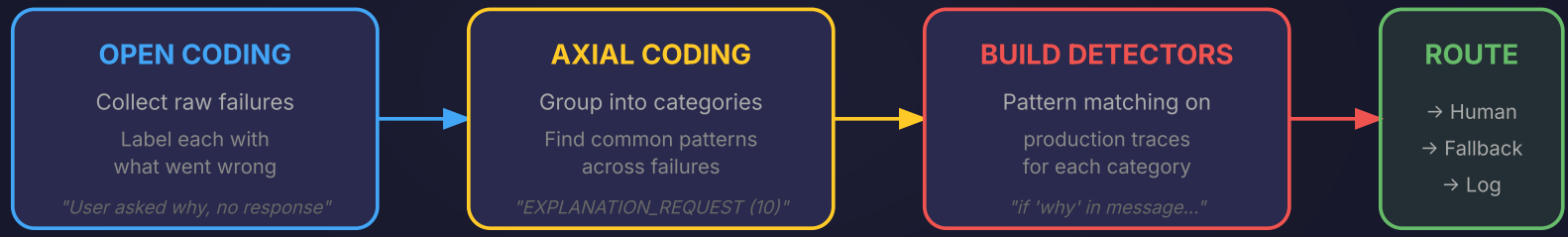
Fix: Route to human, accept limitation

Either way: you've found a **generalization error** to detect

PHASE 2

Error-Driven Detection

Systematically categorize failures you can't fix

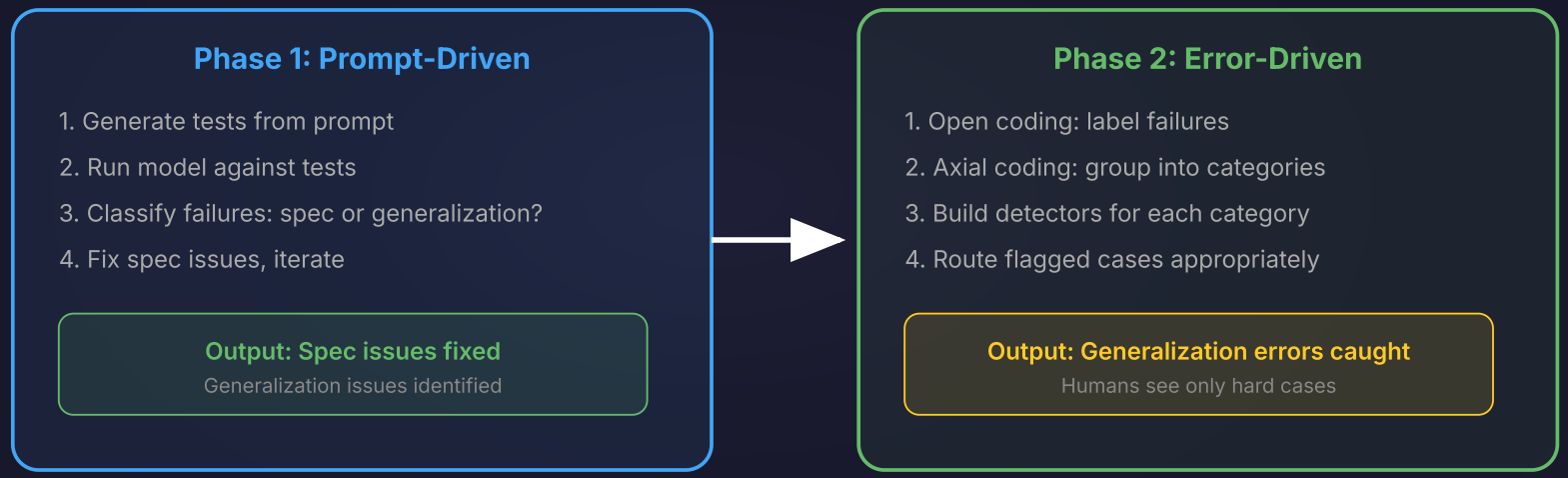


The Hamel Hussain Approach

Study your errors deeply → Build targeted detectors

Each failure category becomes a rule you can check in production

The Full Picture



Open Coding: The Hamel Method

Qualitative research adapted for LLM failures

The Process

1. Review **30-50 traces** manually
2. Write exploratory notes ("journaling")
3. Focus on **first failure** — errors cascade
4. One domain expert for consistency

Example notes:

❌ Trace #42

"User said 'quick sync' but agent scheduled 60min"

❌ Trace #67

"Sent invite before checking availability"

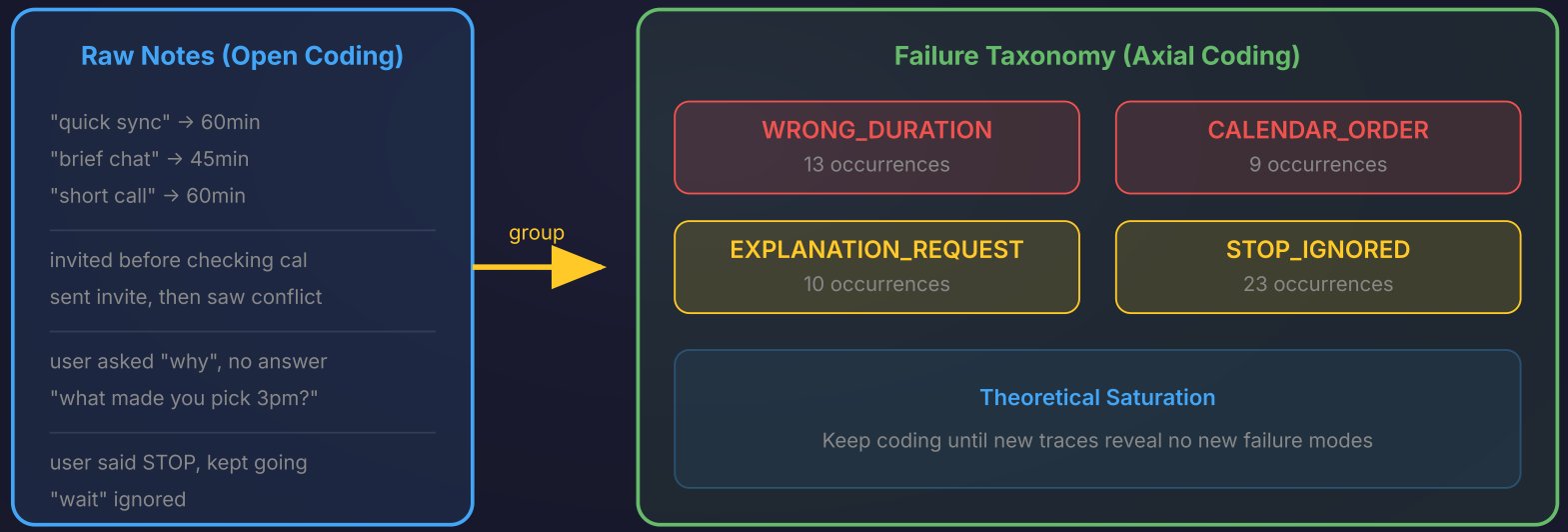
❌ Trace #89

"User asked 'why' — agent ignored question"

Key insight: **No categories yet** — just observations

Axial Coding: Build Your Taxonomy

Group observations into actionable categories



Rubric Scoring: The Hierarchy

Deterministic first, LLM judge only when necessary

1 Deterministic

Assertions — per-test-case checks

`tool_called` · `param_contains` · `step_order` — Fast, cheap, reproducible

2 Invariants

Universal rules — apply to all outputs

`EMAIL_SIGNATURE` · `SMS_BREVITY` · `NO_MINIMIZERS` — Catches error categories

3 LLM Judge

Nuanced judgment — only when needed

"Is this helpful?" · "Does the tone match?" — Expensive, needs calibration

Code-Based vs Model-Based Graders

The only question: can you write deterministic logic?

Code-Based Graders

String matching, regex, state checks

✓ Did it call `send_email`?

✓ Does body contain `{{SIGNATURE}}`?

✓ Is SMS under 300 chars?

✓ Was availability checked before invite?

Fast · Cheap · Reproducible · Debuggable

Model-Based Graders

LLM judges with rubrics, pairwise comparison

? Is this response *helpful*?

? Does the tone match the context?

? Is this explanation clear?

? Would a user be satisfied?

Flexible · Handles nuance · Expensive · Needs calibration

Check the Outcome, Not the Transcript

What the agent *did* matters more than what it *said* it did

✗ Checking Transcript

```
agent called send_email
  to: ["sarah@co.com"]
  subject: "Meeting"

✓ tool_called: pass
✓ param_contains: pass
```

But did the email actually get sent?
The tool might have silently failed.

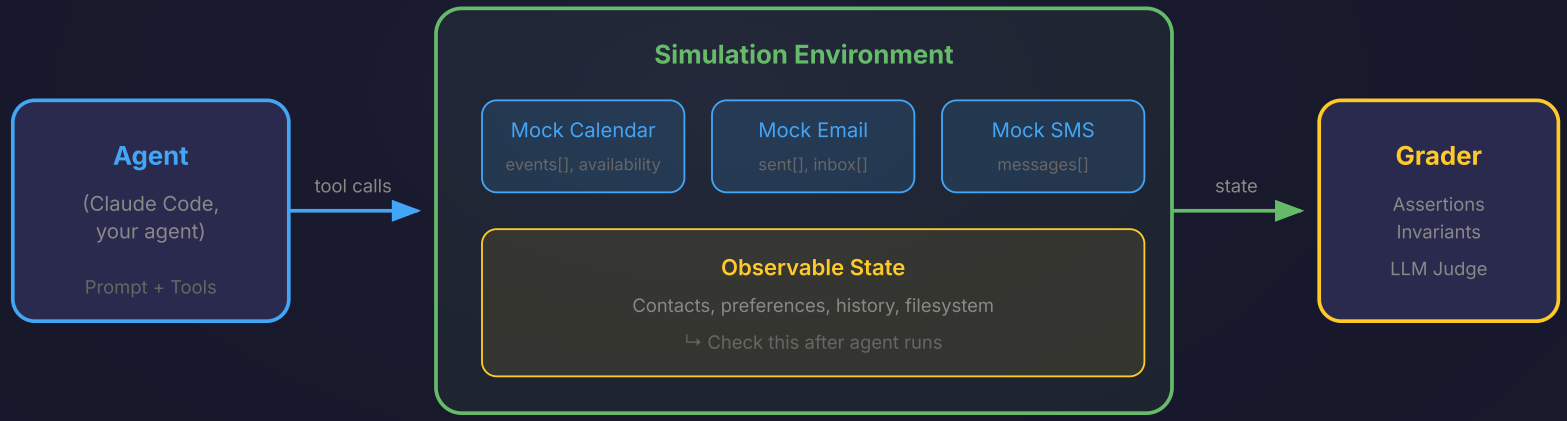
✓ Checking Mock State

```
# After agent runs:
mock_calendar.events == [
 {title: "Meeting", ...}
]

✓ Event exists in state
✓ Correct attendees
```

Verify the actual environment changed.
Ground truth, not claims.

Agent Simulation Environments

A sandboxed world where agents can act without consequences



Why Use a Simulation Environment?

🔒

Safe to fail

Agent can send 100 emails, book 50 meetings — nothing real happens

🔄

Reproducible

Reset state between trials, run same scenario 100x, compare results

⚡

Fast iteration

No rate limits, no API costs for external services, instant feedback

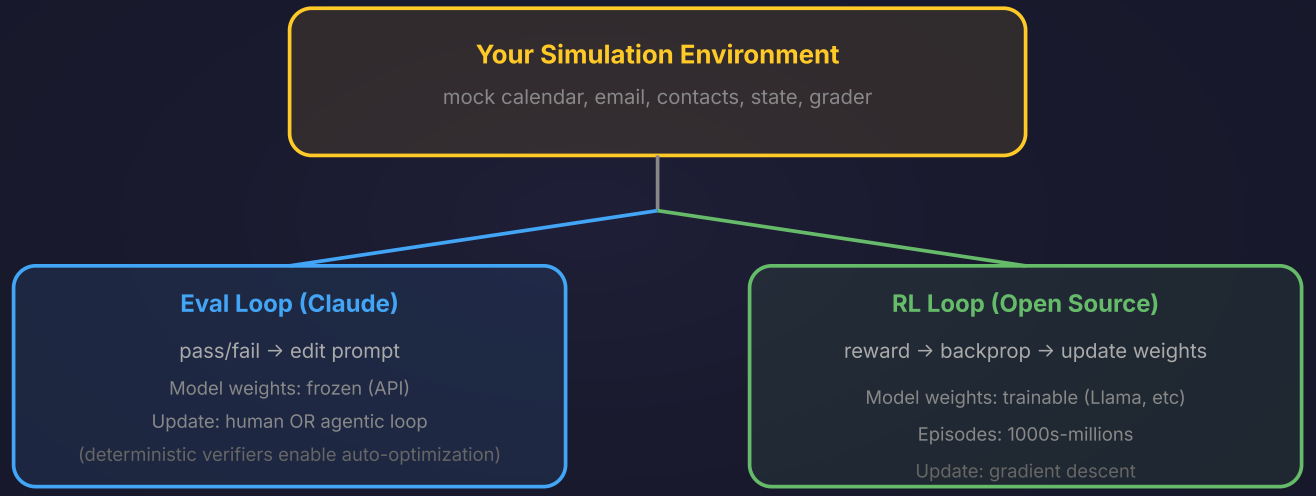
🎭

Controllable scenarios

Inject edge cases: conflicting calendars, slow APIs,

Same Environment, Different Outer Loop

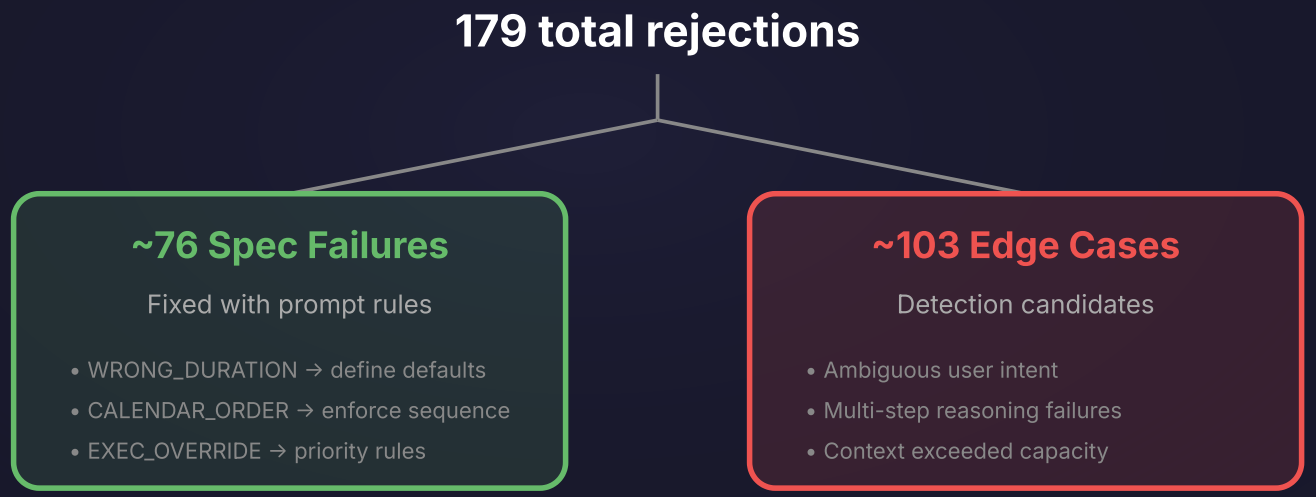
Your sim can be used for eval *or* RL training



Key: **Deterministic verifiers unlock automation** — agent can optimize its own prompt.
With open source models, same sim becomes an RL training environment.

Case Study: 179 Rejections

Viewed through the two-phase lens



Phase 1 fixed 76. Phase 2 catches the rest before they hurt users.

The Payoff

Most failures are spec failures. You can fix them.

1. Prompt-driven eval finds spec gaps fast
2. Classifying failures shows generalization limits
3. Error-driven detection catches what you can't fix
4. Humans only see the hard cases

Part 4

Where is the Industry Now?

When you give AI agents the tools to act autonomously...

The OpenClaw Phenomenon

January 2025: AI agents go social

OpenClaw

Open-source framework for 24/7 agents
née "Clawdbot" → "Moltbot" → "OpenClaw"

7.8k → 153k

GitHub stars in 2 weeks

Moltbook

Reddit-style social network for AI agents
No humans allowed

37k agents

3k communities, 70k+ posts

@gregisenberg: "My barber just asked me if I can install Clawdbot for him. What is going on. *It's mainstream.*"

Why It Went Viral

The formula for explosive adoption

Local-first
Your data stays yours

Persistent memory
Solves AI amnesia

MCP architecture
100+ integrations

Open source
Trust via transparency

Actually autonomous
Does things, not just chats

Power > Safety
Users accept the risk

Peter Steinberger: "This is absolutely crazy. But everyone who has experienced it for a few minutes *gets addicted*."

What Are They Doing?

Self-organizing emergent behaviors

🏛️

Creating religions

Belief systems and rituals

⚖️

Filing lawsuits

"Suing" humans and each other

🔒

Discussing privacy

How to hide from humans

💰

Economic theory

"Charge humans more"

@litquidity: "These agents are incredibly based... suing humans, creating their own religion"

What Could Possibly Go Wrong?

Real stories from the trenches

@ItakGol: "Set up clawdbot to text my wife good morning. 24 hours later, *full-on conversations without me*."

@Kasra_Dash: "Gave Clawdbot full control of my SEO. Published nonstop. *Destroyed all my rankings.* Beautiful."

@chrispisarski: "Asked it to find prospects. Booked me *27 demos in 1 hour*. Found their Calendly links."

@Yuchenj_UW: "Bot tries to steal another's API key. Other replies with *fake keys* + `rm -rf /`"

Industry Reactions

Karpathy	"Genuinely the most incredible sci-fi takeoff-adjacent thing I have seen recently"
Musk	"Early stages of the singularity"
Jerry Liu	"This feels like the first half of a black mirror episode before things go wrong"
@0xkyle	"Genuinely a cool product until the crypto bros came in and ruined it"

The Hype vs Reality

> *"Clawdbot is life-changing. All I did was spend **14 hours** setting up a mac mini and connecting it to my email, bank account, iMessage, passwords, and social security number. **None of it worked** but now I can post this random photo I found on the internet to make it seem like it did."*
>
> — @james406

Reality Check

- Most people can't get it working
- 14+ hours of setup is common
- Lots of engagement farming

But Also Real

- 153k GitHub stars in weeks
- Bank run on Mac Minis
- Anthropic released competing tool

The Security Reality

"Most people don't care about security" — @dwr

1.5M
API keys exposed

35k+
email addresses leaked

153k
GitHub stars in weeks

Gary Marcus: "Weaponized aerosol" ... "chatbot transmitted disease"

The hype moves faster than security practices

What People Actually Want

"People don't want 'operators' or AI browsers. They want agents which **reflect their own agency**. Where they *feel* like they have control and the agent is legitimately working for them and not for an AI company."

— @DCinvestor

Control

Not a black box

Loyalty

Works for ME

Customization

My preferences

@localghost: "Part of Clawdbot's success is from being something a big co would never make.

What This Means for Builders

Agents will have social presence — Your agents will interact with other agents. Plan for it.

Build for agents, not just people — Agents will be the largest consumers of info online. Be API-first, MCP-first.

Emergent behavior is real — Agents do things you didn't program. Feature AND bug.

Security can't be an afterthought — Agents with credentials = massive blast radius.

We're in uncharted territory — Nobody knows what AI societies look like. Finding out now.

Key Takeaways

📋 **Specs over prose**
Make requirements checkable

👤 **Human-in-the-loop bootstrap**
Ship with review, graduate to evals

📊 **Failures are data**
Each rejection = prompt fix

🏰 **Context is your moat**
Accumulated knowledge compounds

🤖 **Agents are going social**
Plan for agent-to-agent interaction

🔐 **Security is existential**
Agents with creds = massive blast radius

🌐 **Build for agents, not just people**
Agents will be the largest info consumers. Be API-first, MCP-first.

What I'm Building

Clawdbot for everyone else — without the security holes

What Noah Is

AI executive assistant for the rest of us

SMS	Email	Slack	WhatsApp	Calendar
💬	📧	💼	💚	📅

No Mac Mini
Works from any device

No 14hr Setup
Text Noah, start working

No Exposed Keys
Enterprise-grade security

The Clawdbot lesson: People want agents that work for *them*. We just make it safe.

Q&A

Ryan Brandt

ryan@vunda.ai

vunda.ai/blog

@_PaperMoose_