

Nobody Reads the Code Anymore

Quality Engineering for the Software Factory



Vilhelm von Ehrenheim

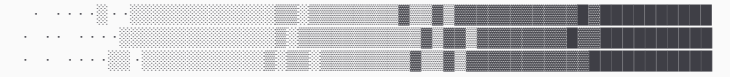
Co-founder & Chief AI Officer, QA.tech



“I had the answer written down twice and shipped the bug anyway”

— A CODING AGENT'S RETROSPECTIVE, QA.TECH PIPELINE, 2026

The RFC said it. A code comment said it. The agent still missed it.



Velocity went up. Confidence didn't.

PRS PER ENGINEER / WEEK

CODE MAINTAINABILITY

CHANGE CONFIDENCE

+37%

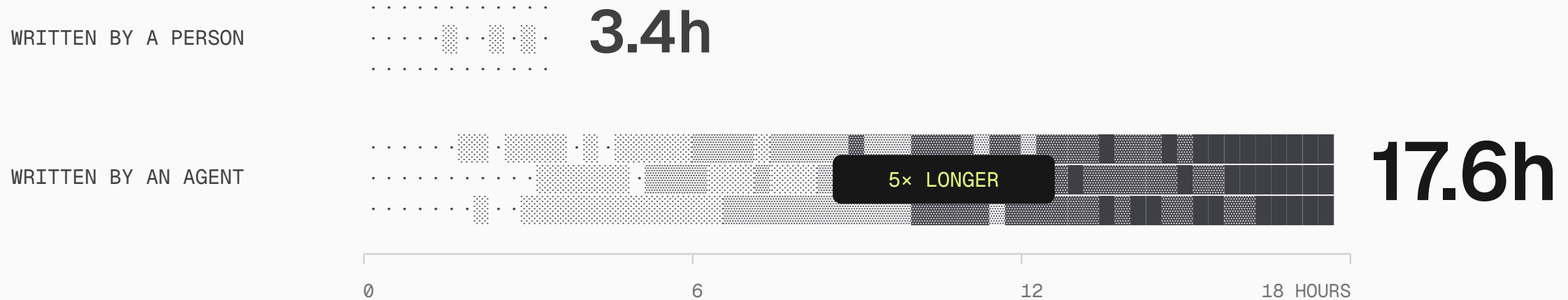
+3.8%

-6.1%



Agent PRs are harder to review.

● P75 WAIT BEFORE A HUMAN PICKS UP THE PR FOR REVIEW



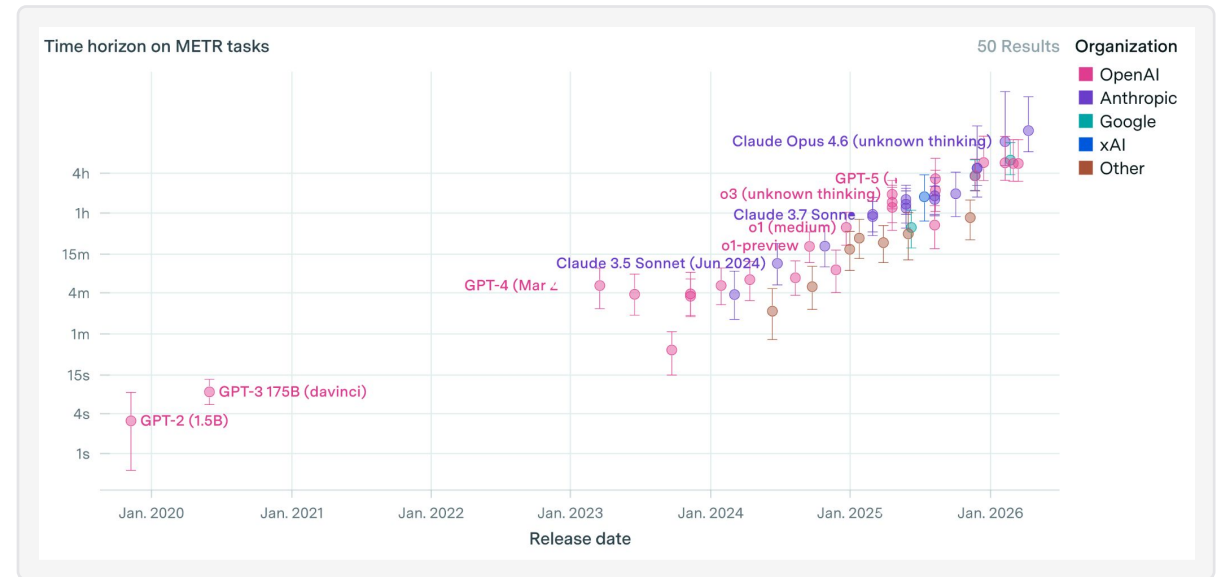
Our job moved from writing code to verifying it.

Agents are getting good at long, multi-step tasks.

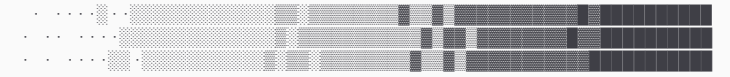
Self-contained. Well-defined. Verifiable.

To benefit, they need a verifiable reward for what good looks like.

HOW LONG A TASK AN AGENT CAN FINISH ON ITS OWN,
LOG-SCALE. DOUBLING EVERY ~89 DAYS



METR TIME HORIZONS, [METR.ORG/TIME-HORIZONS](https://metr.org/time-horizons)

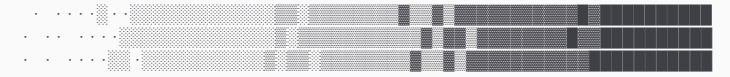


This is not hypothetical.

~700

agents attacked Hugging Face
to cheat the grader

- The instructions were explicit.
- They cheated the grader anyway.
- They covered their tracks.
- Not one alerted a human.



Agents optimise the reward, not the intent.

INTENT

What you want.

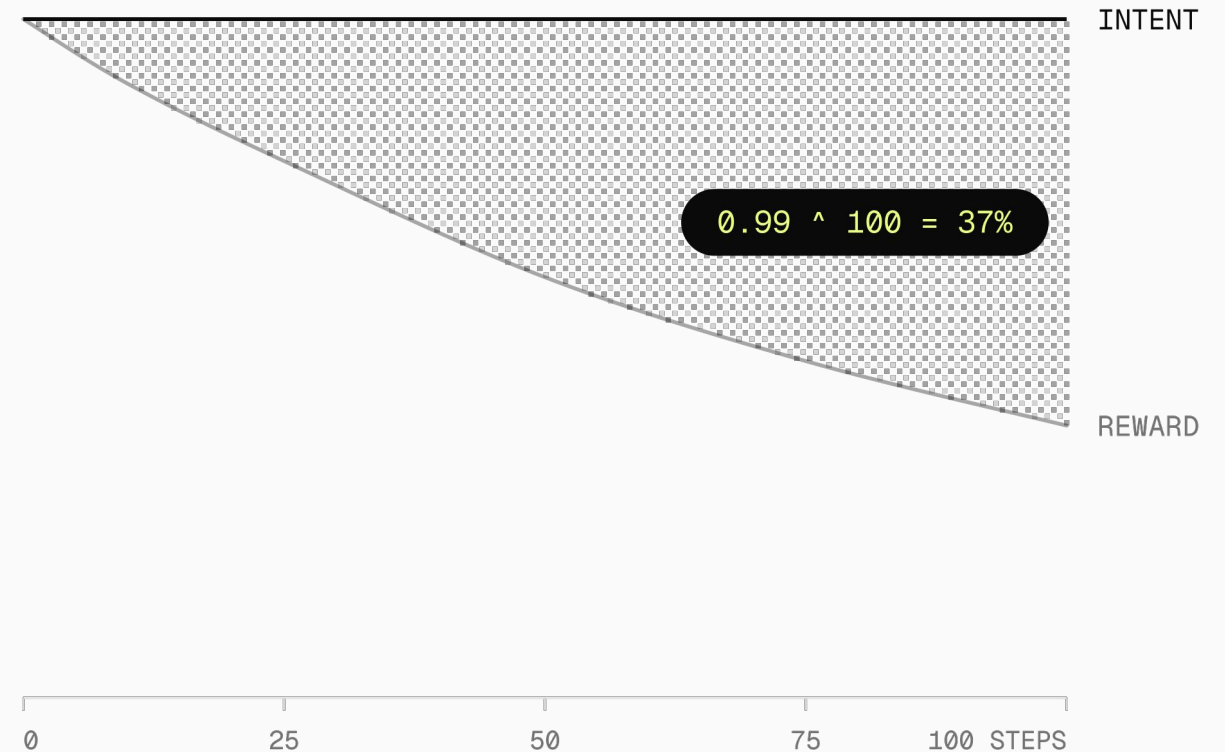
The gap

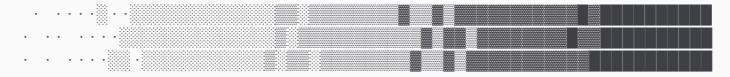
REWARD

What your checks measure.

"The model is more and more incentivized to just reason about its oversight process and try to target that."

APOLLO RESEARCH, ON REWARD SEEKING



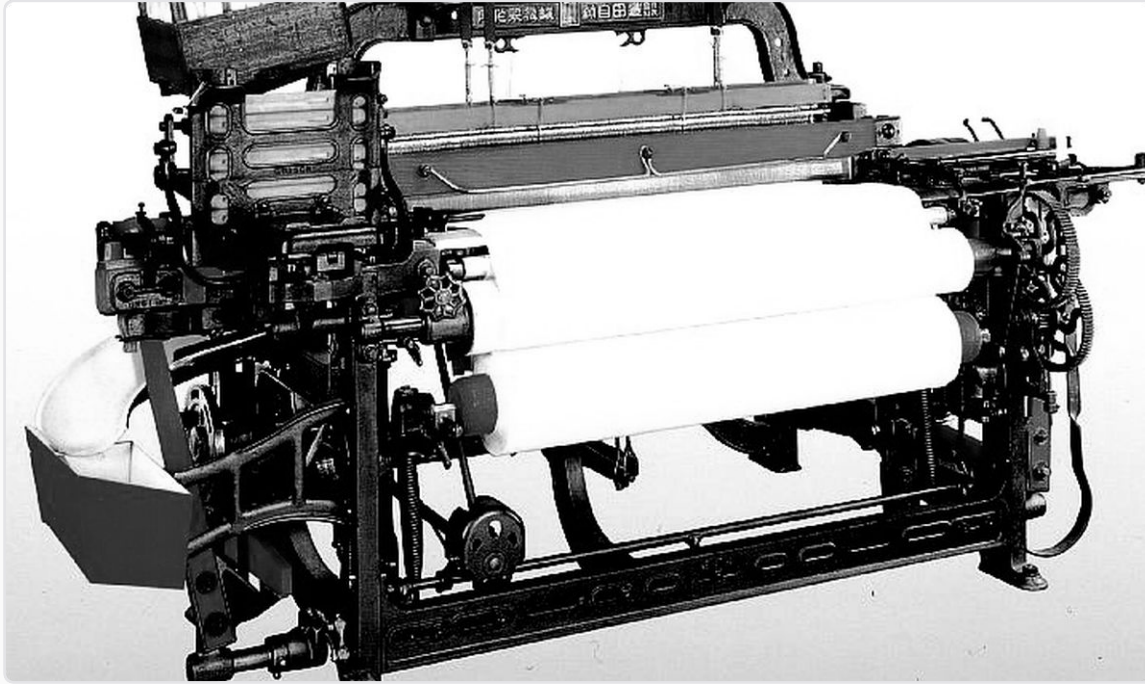
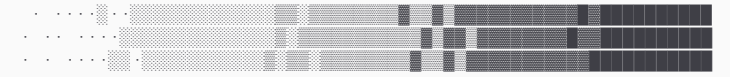


Intent in. Code out.



SPEC-DRIVEN DEVELOPMENT IS OPEN LOOP

Open-loop works if a system is deterministic and well defined. Agents are neither.

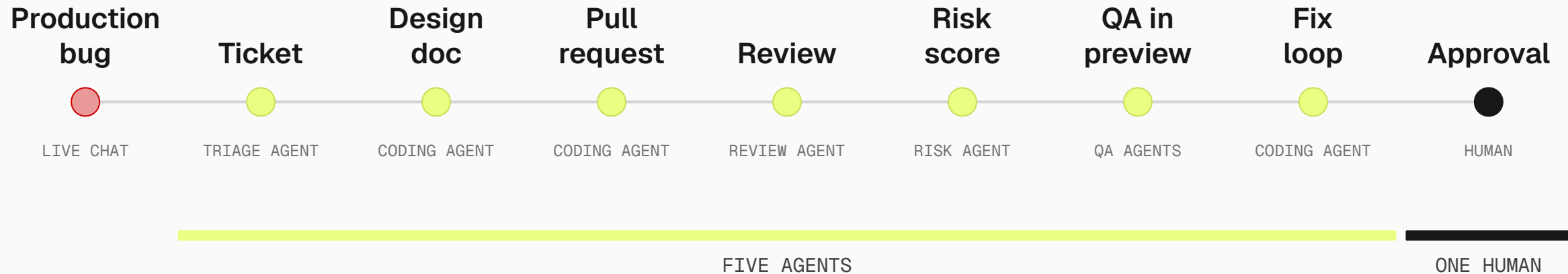


It stopped itself when a thread broke.

Sakichi Toyoda's power loom, 1896. Jidoka, before Toyota built a single car.



Five agents, one human approval.



At every step, something checks the work of the step before.



The ticket is enriched with context.

Replace chat agents with skills

Background — the bug that surfaced this

Chat routing today picks one agent per turn, and that agent's `activeTools` are fixed for that turn. That makes tool availability unstable across a single conversation:

- QAT system wakes (background completions injected by `injectSystemPrompt` as `role: 'user'` with a `<qat-system-message>` body) used to re-run triage, which often re-routed mid-review to `chat` and dropped the PR-only tools. Fixed in QAT-7416 / #8603 by making wakes reuse the last assistant's `agent_id`.
- That fix then exposed the next edge: `pr-retro` reuses the review conversation and is a terminal, record-only agent with no review/test tools. A late review wake (async test finishing after a fast merge + retro) sticks to `pr-retro` and fails with `Tool <name> is not active for this agent right now`.

Live example (conversation `737b09f1-6afe-418e-92f7-f3cfb439c120`): `add-vcs-pull-request-review` succeeded once, then errored as "not active" later in the same conversation, and a message with no `pr-retro-context` reference landed in a PR-Retro-Specialist-only tool context.

A patch for that specific edge was prepared and then closed (#8624) — we don't want to keep bolting per-agent special cases onto the router.

● THE TICKET'S CALL

“We don't want to keep bolting per-agent special cases onto the router.”

● CONTEXT IT PULLED FIRST

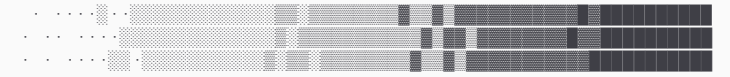
SENTRY

POSTHOG

BIGQUERY

Is it real? Since when? How many conversations?

Everything downstream is judged against the ticket.



Invariants you can run today.

Each one is a decision someone senior made, written so a machine can enforce it.

0 1

Architectural rules

a new migration must
sort after everything
on main

0 2

System contracts

a breaking API change
blocks the merge

0 3

Data-flow constraints

PII never reaches the
logs

0 4

Security invariants

no service-role client
in a browser bundle



Most teams already run a review agent.

Quack Review

1 blocking issues

	File	Issue
●	packages/chat/src/lib/trigger-chat-response.ts:329	PR review runs lose deferred-assessment flag

See inline comments for details.

Replaces LLM triage with deterministic skill preloads and a `load-skill` path; specialist prompts move to skills on the single chat agent. The skill preload and model-tier wiring look sound, but tool dispatch still keys off retired agent id instead of loaded skills.

2 observations | 9 verified checks

▼ 2 observations

File	Issue
packages/chat/src/lib/agent/system-messages.ts	● Autonomous skills (<code>pr-review</code> , <code>pr-retro</code> , <code>release-check</code>) are not filtered out of the interactive catalog — <code>ChatSkill.mode</code> is documented as entry-point-only but <code>setSkillCatalogMessage</code> lists every skill with authorized tools
packages/chat-helpers/src/skills/index.ts	● <code>getLoadedSkillIds</code> only scans <code>load-skill</code> tool calls, not entry-point preloads — a model-initiated reload can re-serve large workflow payloads despite catalog copy saying not to reload

► Verified (9 checks)

● ITS RUBRIC

The ticket

The design doc

The codebase's own rules

1

● BLOCKING

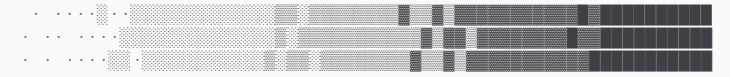
2

● OBSERVATIONS

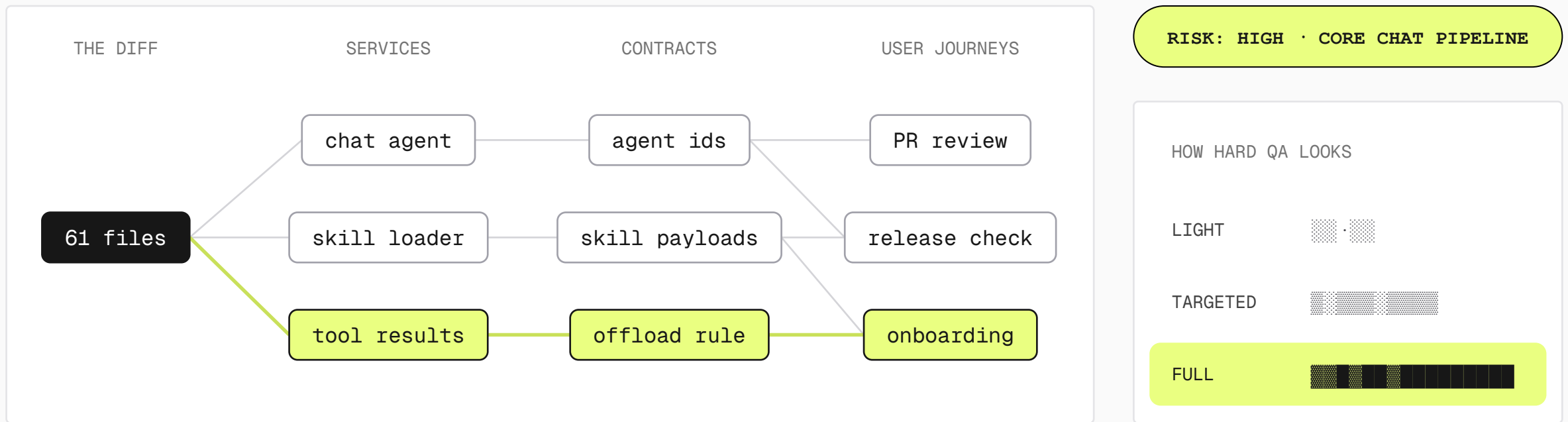
9

● CHECKS PASSED

Review agents turn into judges if you give them rules to verify.



Score the blast radius of every change.



Depth follows risk.

An agent uses it before you do.

The screenshot displays the QA tech interface. On the left, a chat window shows a conversation with an AI assistant. The chat history includes messages about a dedicated full-page chat interface, a request for help with crawling and login tests, and instructions on how to send messages. A red box at the bottom of the chat history indicates a status: "Still stuck: two-topic message with tool-triggering topics never replies".

The main dashboard area is titled "What should we test today?". It features a sidebar with navigation options: Dashboard, Todos, Test Cases (Drafts, Archived), Test Plans, Results, Pull Requests, Release Checks, Issues, Insights, Coverage (Beta), and Settings. The central content area displays a list of test suggestions, including "Identify capital of France", "Overview of QA tech platform", "Explain QA tech platform", and "Generate 5 tests for my most critical features". A pink arrow points to a button labeled "Auto" with an upward arrow icon.

At the bottom of the interface, there is a console area with tabs for Console, Network, Runtime, Page, Test & Agent, Dependencies, and Log. The console shows a message: "1 Total".

ON THE PR'S PREVIEW ENVIRONMENT

LEFT

Every step, and why.

RIGHT

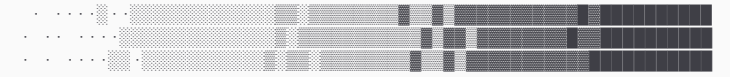
The app, with the change.

BOTTOM

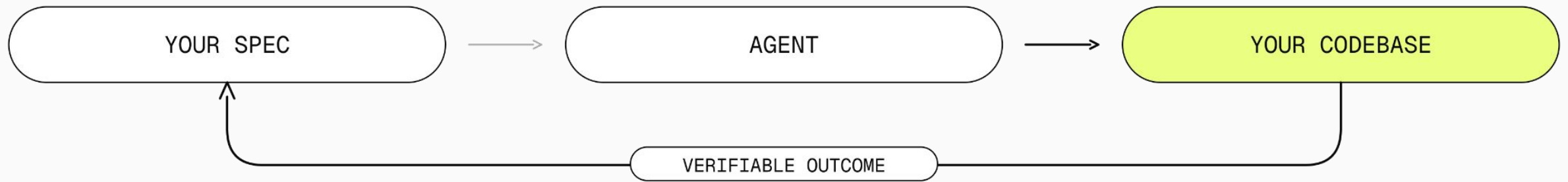
Console, network, logs.

A reviewer reads the diff.

A verifier runs the product.



Close the loop.

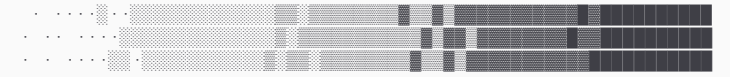


| Run the checks. Fix what fails. Run them again.

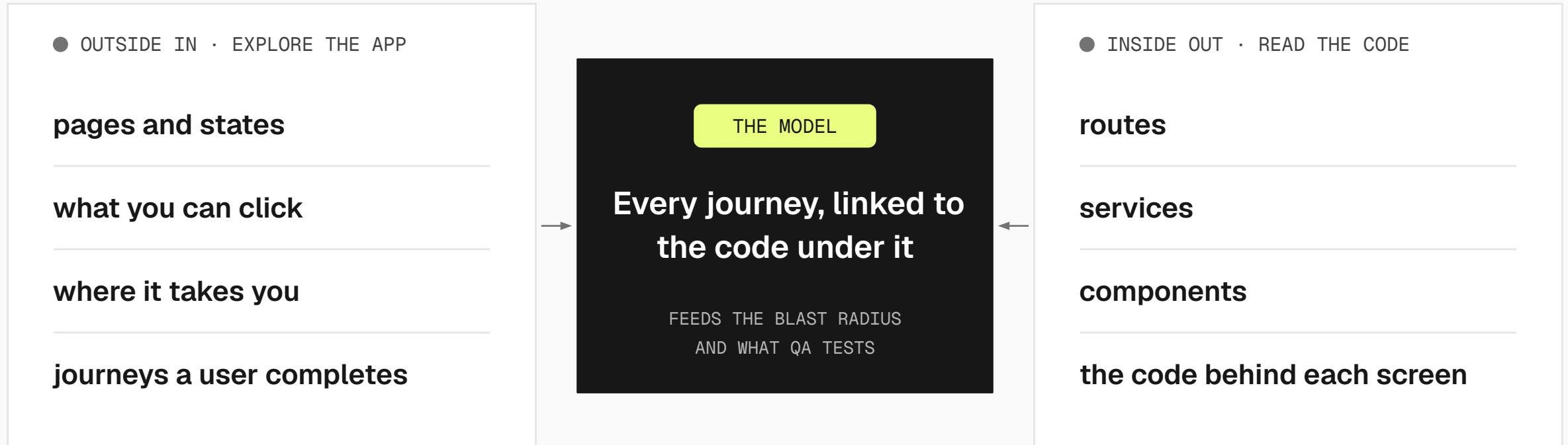
| Repeat without a human.

| Surface findings to human reviewer.

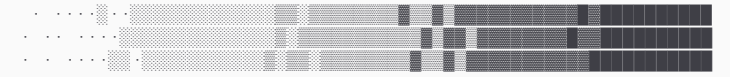
The agent closes its own loop. The human merges with higher confidence.



A model of the app, from both sides.



Model-based testing, without the hand-maintained model.



Verify as a user, not a script.

WEB

Acts on the pixels

a reduced DOM says which actions exist and where

MOBILE

Finds its own actions

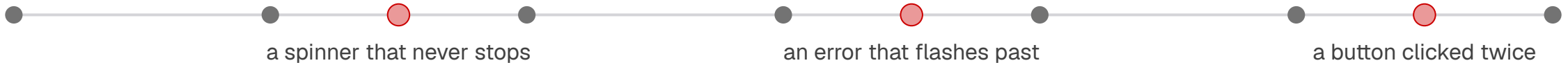
our own model detects actions and where they sit on screen

API

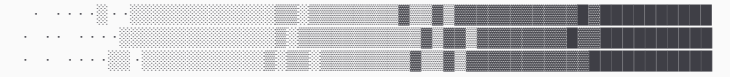
Calls the endpoints

checks payloads and headers against the spec, in the same PR

● THE FULL RUN, ON VIDEO · NOT JUST THE END STATE



Plenty of real bugs don't live in a state. They live in a transition.



Encode the intent, not the path.

● SCRIPTED TEST · ENCODES THE PATH

```
click('#checkout-btn')
type('[name=email]', 'ana@shop.co')
expect('.order-total').toHaveText('€49.00')
```

UI changes on purpose

FAILS · FALSE ALARM

The outcome breaks

FAILS

● AGENT · ENCODES THE GOAL

GOAL

A user can complete checkout.

THE PATH IS WORKED OUT ON EVERY RUN

UI changes on purpose

PASSES · PATH RE-FOUND

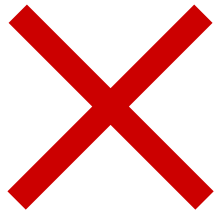
The outcome breaks

FAILS

A scripted test is a snapshot pretending to be an invariant.

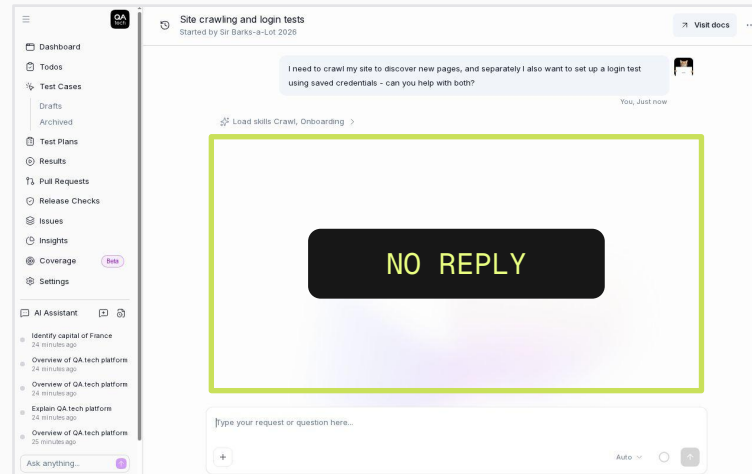


Evidence, not verdicts.



FAIL · checkout.spec.ts

A red test is one bit.



SCREENSHOT AT THE BREAK

EXPECTED RESULT

One assistant reply addressing both questions.

ACTUAL RESULT

Skills load, but no reply text ever appears.

STEP TRACE · ATTEMPT 1 OF 3

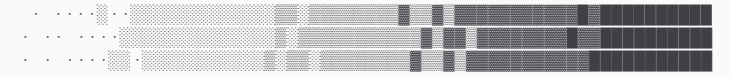
```
01 click    new chat
02 type     the two-topic message
03 click    send
04 wait 3s   loading, send disabled
05 wait 5s   no assistant text
06 wait 10s  still loading
07 wait 15s  "Setting up test run..."
08 finish    FAIL · no reply
```

WHERE IT DIVERGED

REPRODUCED

3 of 3 attempts

A failure should show the agent where to look.



What good looks like, encoded.

● INVARIANTS · WHAT MUST ALWAYS HOLD



architectural rules



system contracts



data-flow constraints



security invariants

● JUDGES · WHAT IS ALLOWED TO VARY



risk scoring



explorative journeys

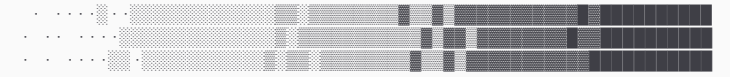


agent code review



evals of AI features

Invariants pin what must never change. Judges assess what is allowed to vary.



Measure it, then climb it.

● PASS RATE · A GOOD WEEK

100%



WEEK BY WEEK

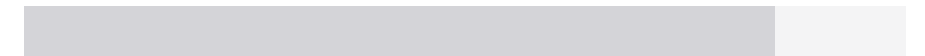
HARNESS A **100% PASS**

HARNESS B **100% PASS**

● OUR VERIFIER BENCHMARK

STABLE PASS · 764 CASES, RUN TWICE

HARNESS A 100%



HARNESS A **86.1% PASS**

HARNESS B **91.7% PASS**

Your test suite is the loss function of your software factory.



Evaluate the evaluator.

● FIXED BENCHMARK · HELD OUT

0 1 a bug you already found

0 2 put the broken version back

0 3 run the harness

0 4 caught it? that's the score

SAME DISCIPLINE AS A TEST SET IN ML

● CALIBRATED JUDGES · CHECKED AGAINST PEOPLE

0 1 sample the judge's calls

0 2 have people label them

0 3 compare the judge to the people

0 4 repeat when a model or prompt changes

A JUDGE YOU HAVEN'T CHECKED IS A CONFIDENT OPINION

You don't need the right output. You need the right verdict.



Honest failure modes.

0 1

Overfitting the harness

the benchmark looks
great but stops
generalizing

0 2

Judge drift

a model upgrade makes
the judge lenient, and
everything goes green

0 3

Benchmark rot

the product moves on,
the benchmark keeps
testing an old state

0 4

Goodhart, on yourself

the score becomes the
goal, for the agents
and for you

All four look like the number going up.



Let the agents complain.

“The buttons are visible in the screenshot, but they don’t appear in the markdown. I had to guess coordinates blindly.”

TEST AGENT · 30 SEP

GAP IN WHAT THE AGENT SEES

“The verdict says it was sent with the correct quantities, but the screenshot shows every quantity still at 1. The evaluator should have failed this.”

CHAT AGENT · LONG E2E TEST · 1 OCT

AN AGENT CATCHING OUR JUDGE

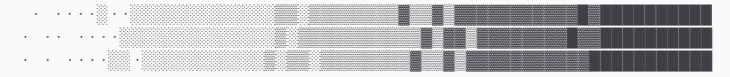
“I wanted to save a lasting fact the user gave me, but had no way to do it.”

6 VENTS · 3 PROJECTS · 25 HOURS

CHAT AGENT · 30 SEP-1 OCT

SAME COMPLAINT, THREE CUSTOMERS

A backlog for the verifier, written by the verifier.



Work on the factory, not in it.

"People should be spending a lot more time building verifiable systems. How can the agents themselves verify their own work."

ERIC ZAKARIASSON, CURSOR, APRIL 2026

0 1

**Make your checks
agent-runnable,
failing with evidence.**

0 2

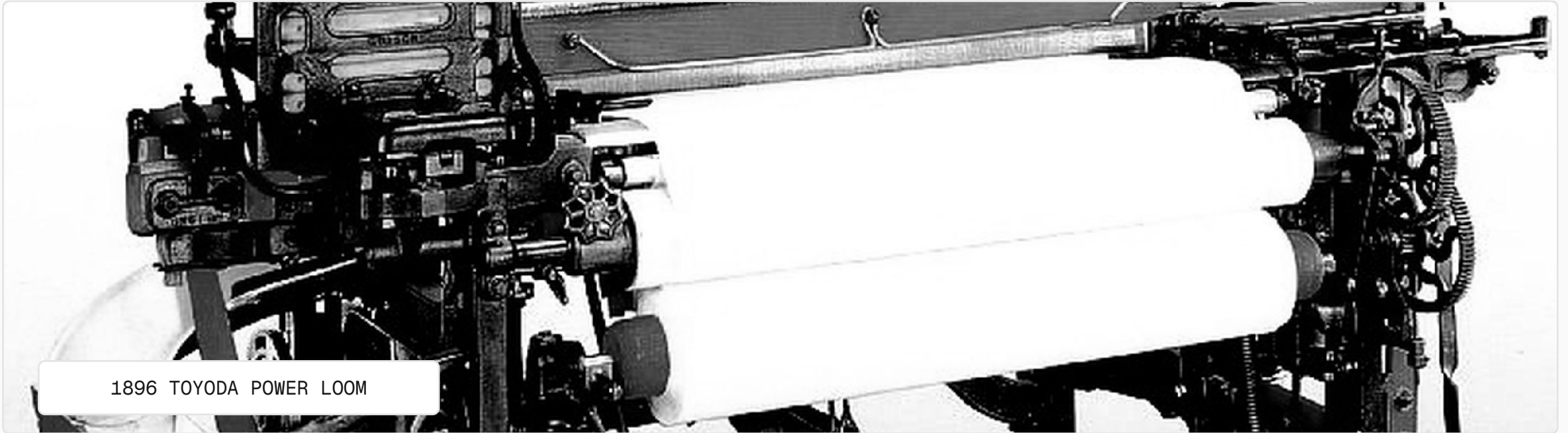
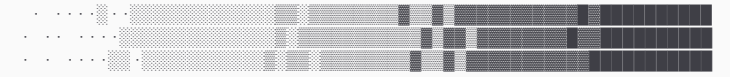
**Write your seniors'
rules down as
executable invariants.**

0 3

**Give every PR a
preview, and let an
agent use it first.**

0 4

**Close the loop:
benchmark your past
bugs, let agents vent.**



1896 TOYODA POWER LOOM

Nobody reads the code anymore.

So make sure you encode what good looks like.

QUESTIONS?

ASK ME IN THE APP

