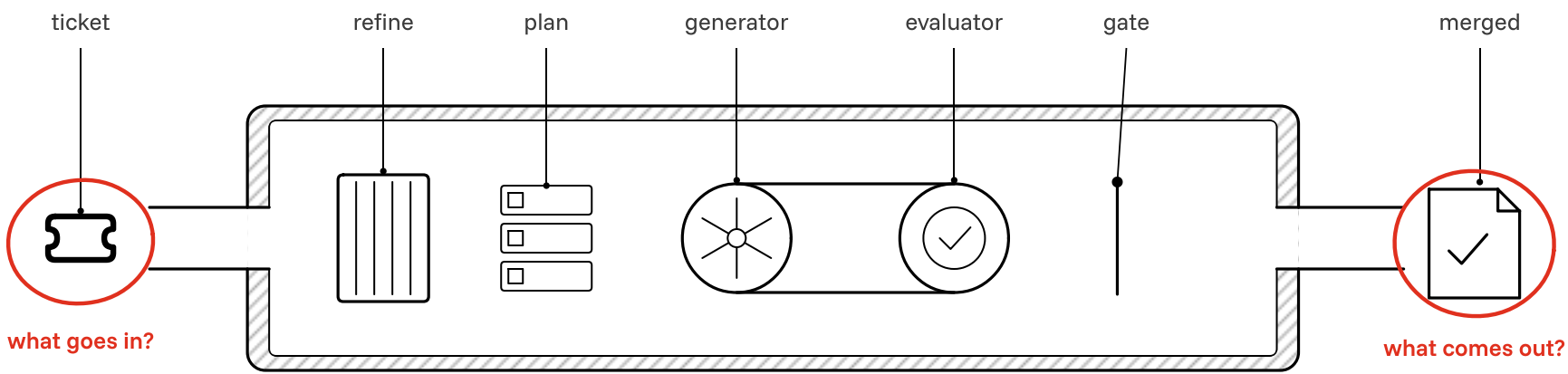
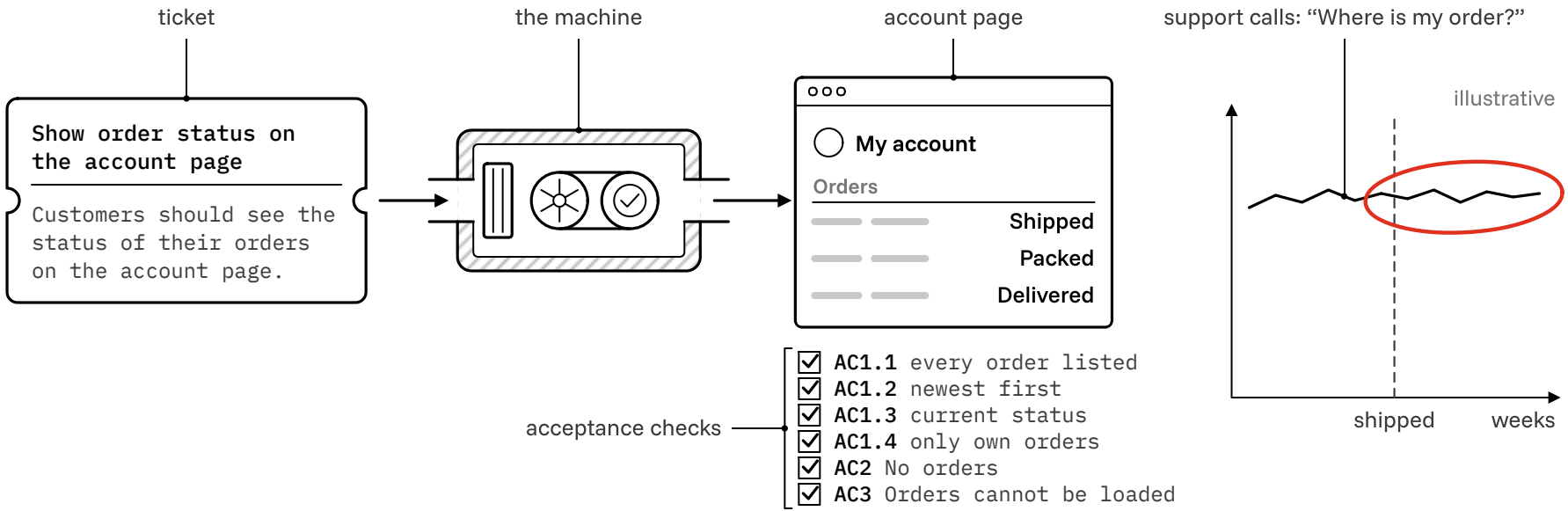
Writing code is **solved.**



A real ticket. The machine. Suppose it ships.

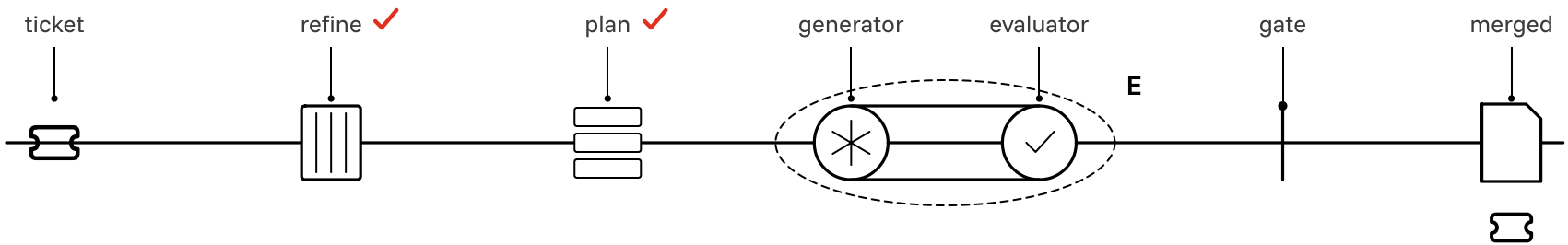


Built exactly as specified. Every check green.

So why are they still calling?

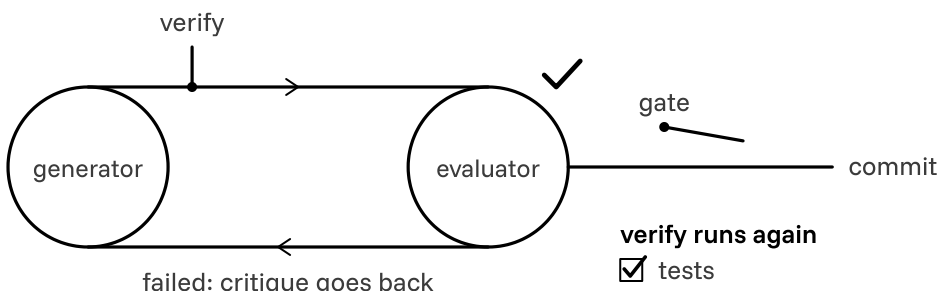
Someone in this community wrote down why. **In 1995.**

One ticket through the machine



DETAIL E · generator and evaluator: the loop

round 2 of 5 · attempt 1 of 3



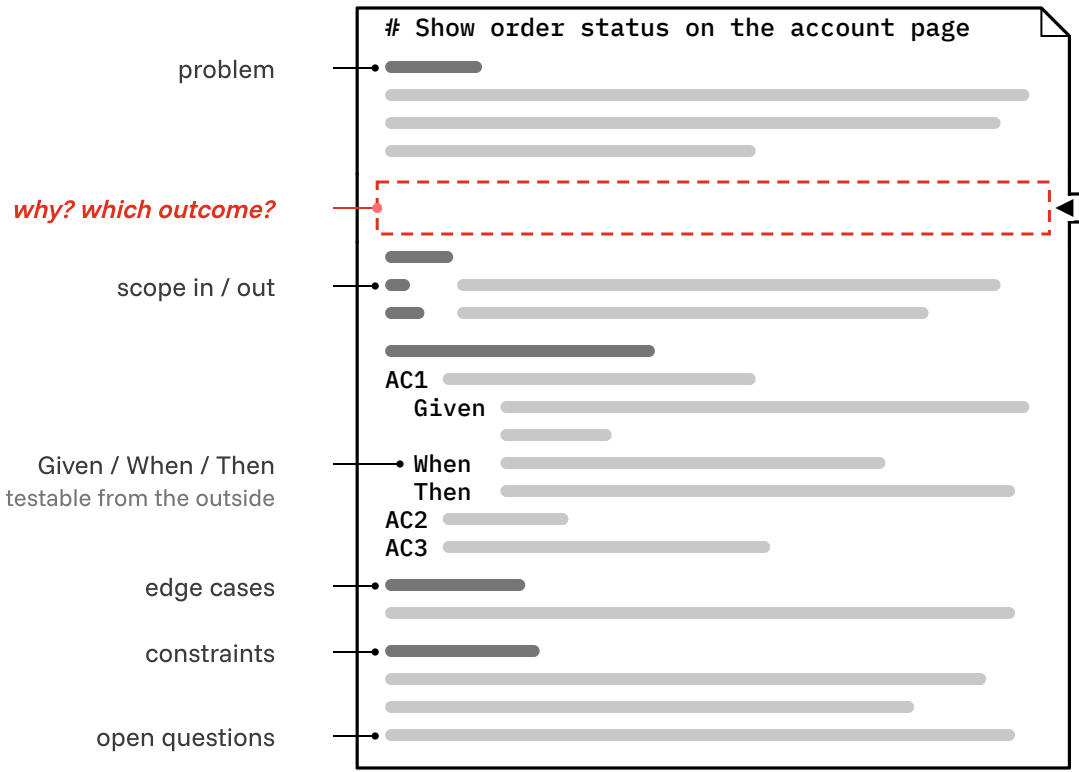
verify runs again

- ☑ tests
- ☑ typecheck
- ☑ lint

- **Generator** writes the code. No deleting a test to get green.
- **Gate**: tests, typecheck and lint run again. Red never commits.
- **Stuck** for 3 rounds: it escalates instead of spinning.

A person approves what to build. The machine has to prove that it built it.

What the machine needs from us



6 questions from refine.
All about the what.

0 about why, or which outcome.

the real refine run for this talk

What we owe the machine

Intent
a business outcome

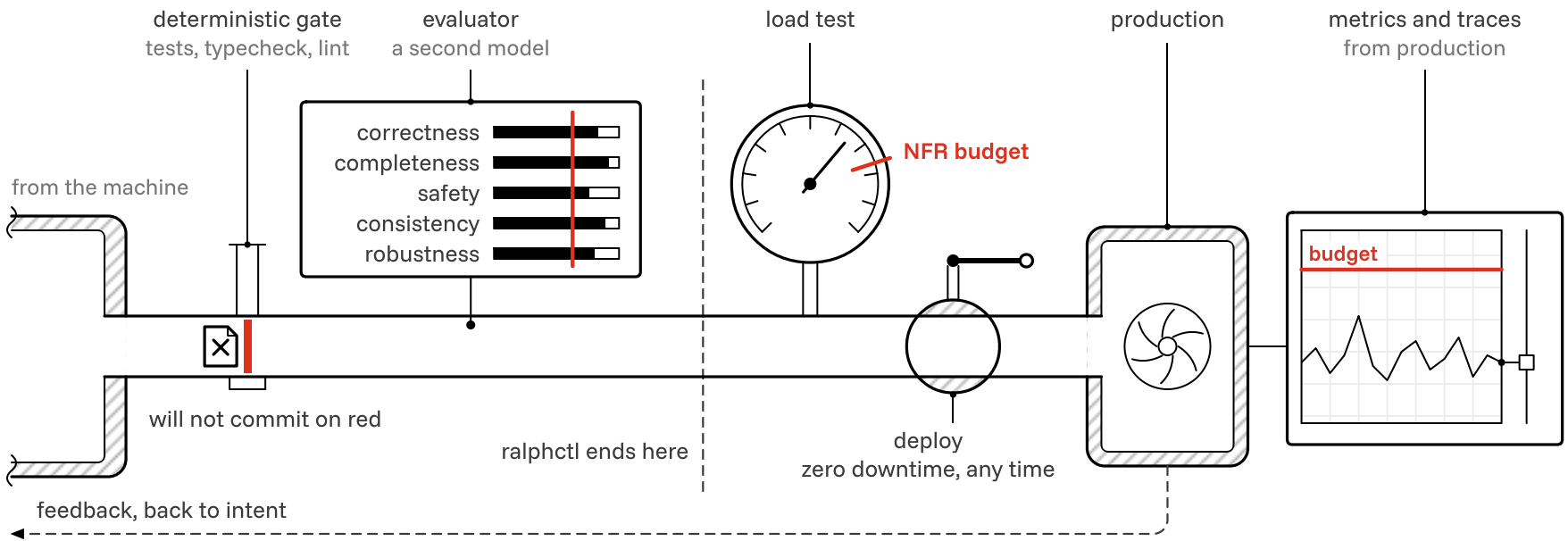
The goal behind the goal
ask why until you reach a business outcome

Context
who, where, which rules

NFR budgets
how fast, what feedback

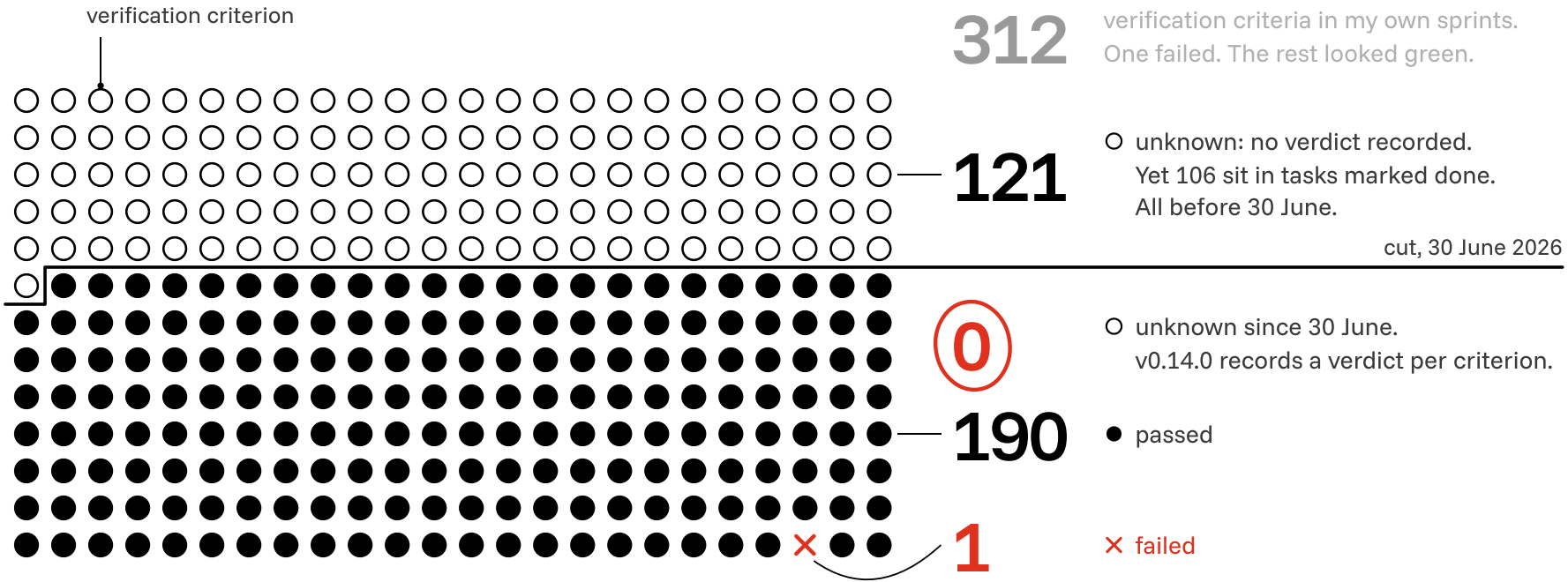
The machine builds what is written. What is written has to carry the why.

What earns my trust



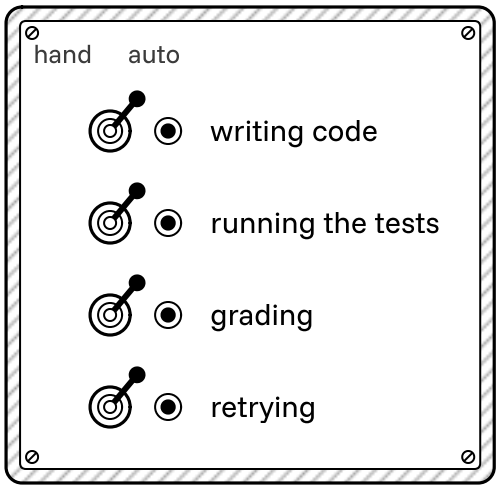
I don’t trust the output. I trust the checks it has to pass. People set the bar.

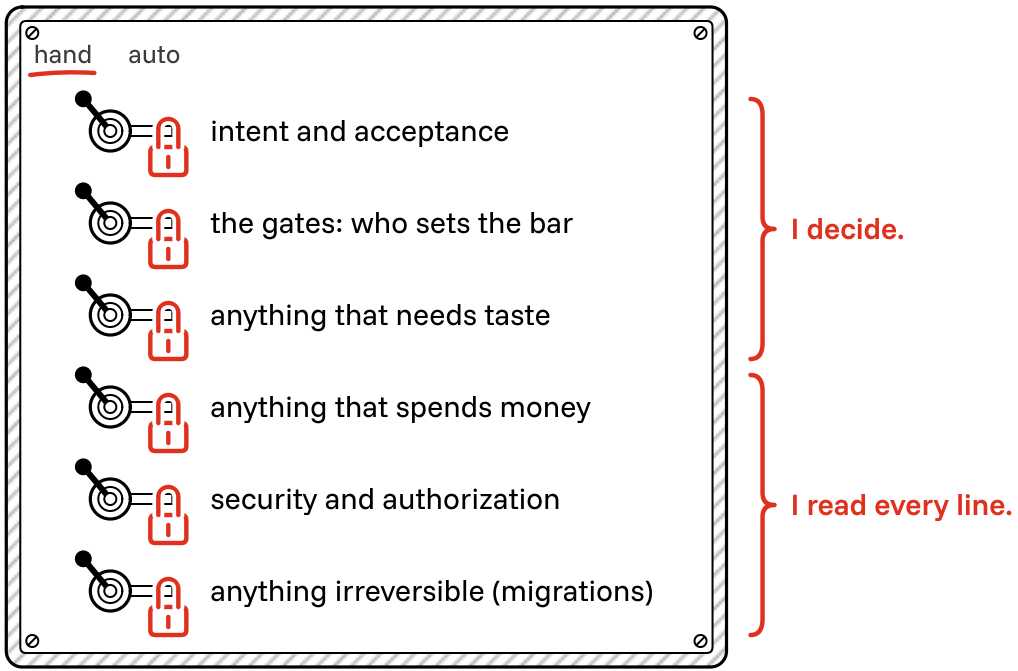
Green is not evidence



“Unknown” is not “passed”. Trust the recorded verdict, not the colour.

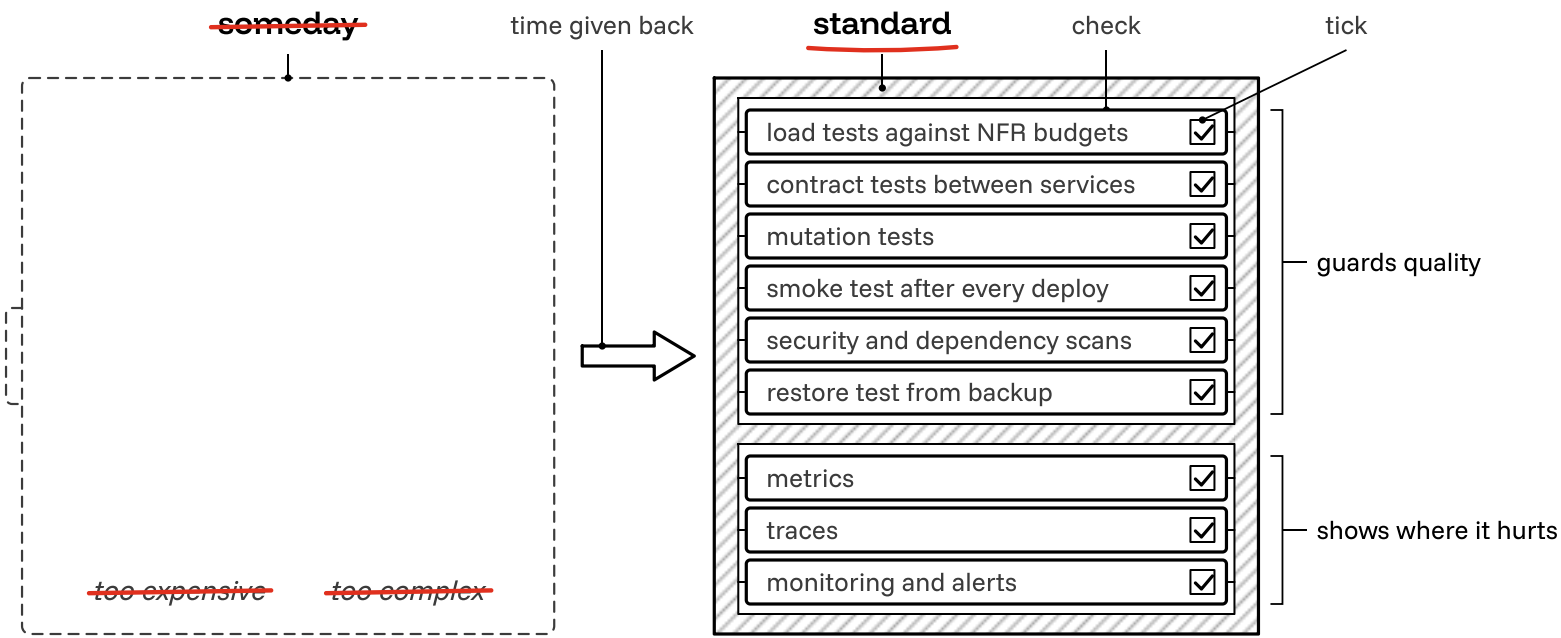
What stays in my hands





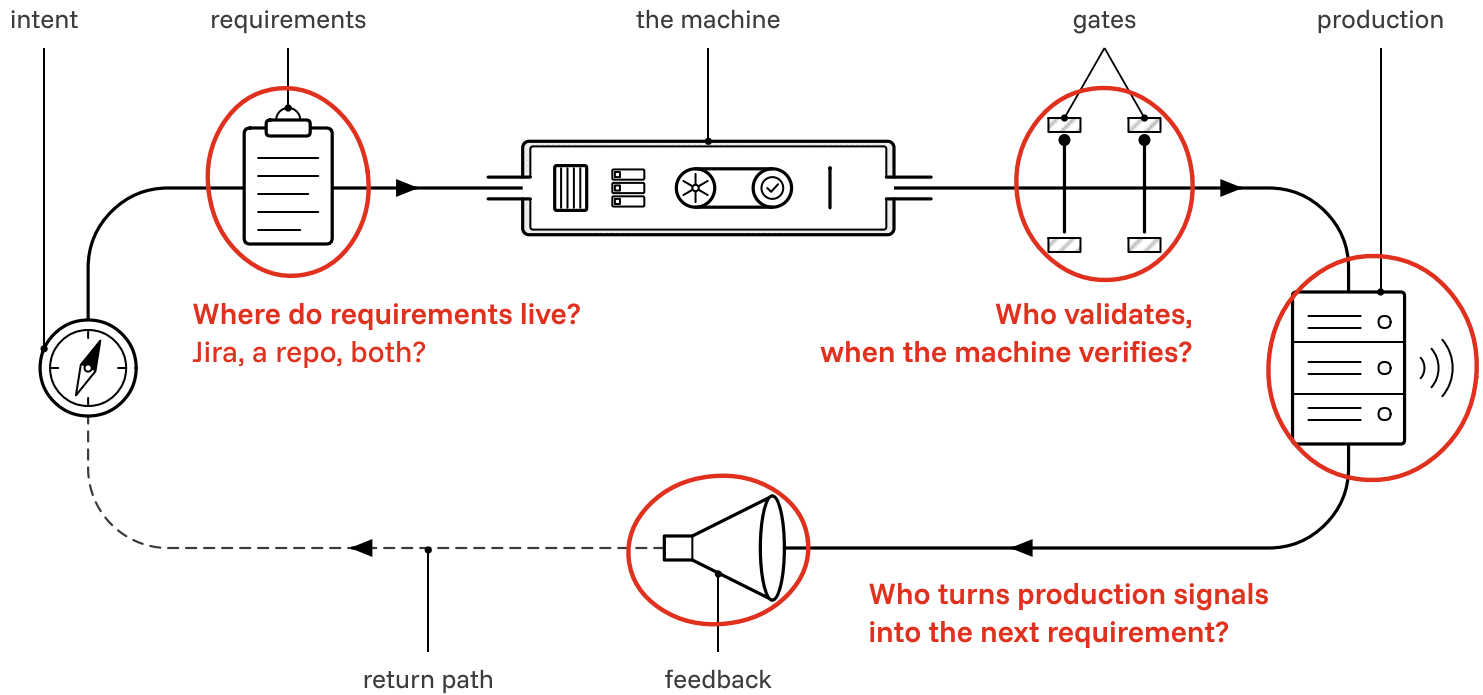
I stopped reading every line. I read the lines that matter.

What we never had time for



The time the machine gives back goes into the checks we always skipped.

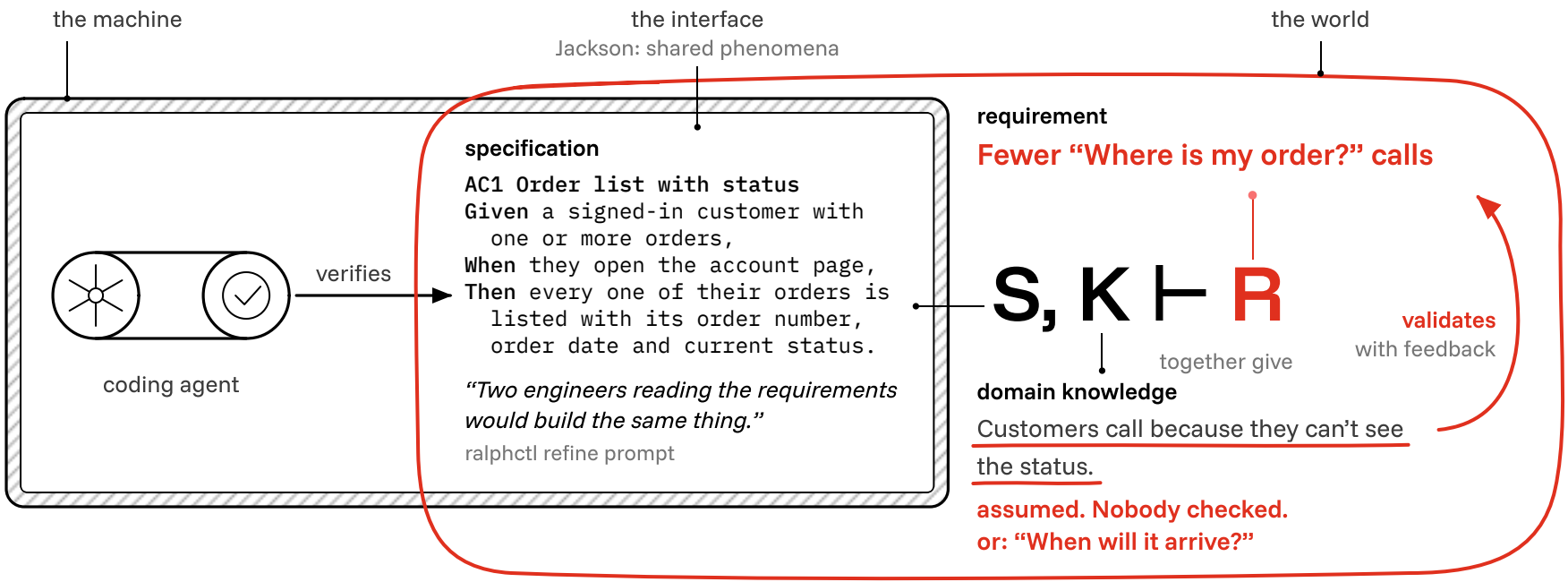
And now?



REFSQ 2027, Basel: **“Aligning RE with AI Velocity”**

The machine is one station. The loop is ours.

Prior art, 1995. Michael Jackson. Not that one.



M. Jackson, The World and the Machine, ICSE 1995 · P. Zave, M. Jackson, Four Dark Corners of Requirements Engineering, ACM TOSEM 1997

Why are they still calling? The machine met the spec. Nobody checked the assumption.

Who has to read your requirements? How? And what do they do with them?

Hands up first: where do yours live today?

☐ a ticket system ☐ a repo ☐ somewhere else

If it's quiet, ask me:

1. Isn't this just waterfall?
2. Which model does what?
3. Can I see it run?

Slides as PDF



Lukas Grigis Lead Software Architect, Zühlke

lukasgrigis.dev · linkedin.com/in/lukas-grigis

github.com/lukas-grigis · `npm i -g ralphctl`

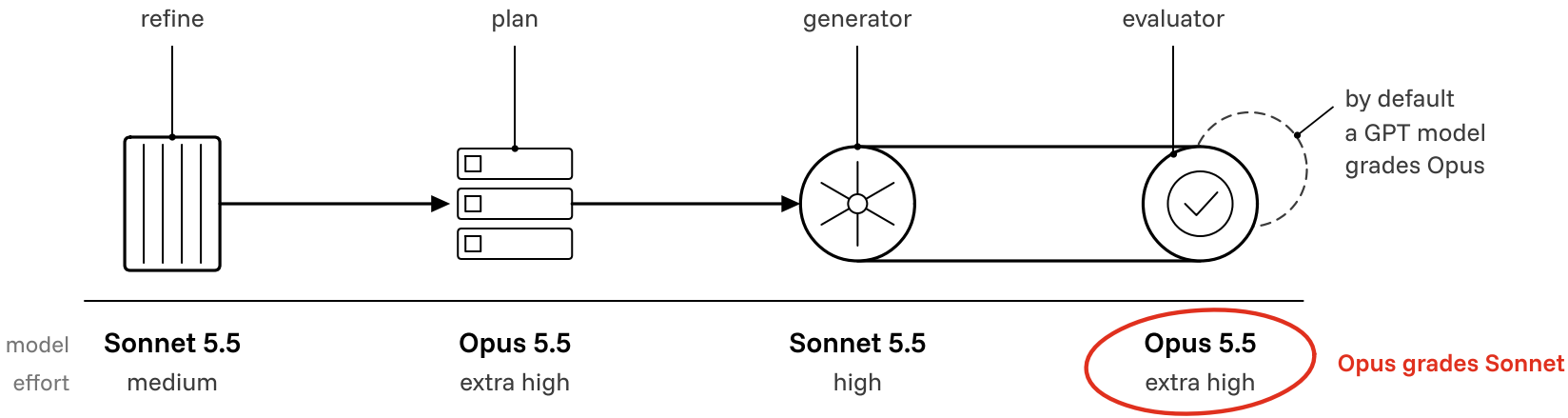
Basel, Switzerland, 12th-15th April 2027
REFSQ 2027
33rd Intl. Working Conference on Requirements Engineering: Foundation for Software Quality

The machine verifies.

We validate.

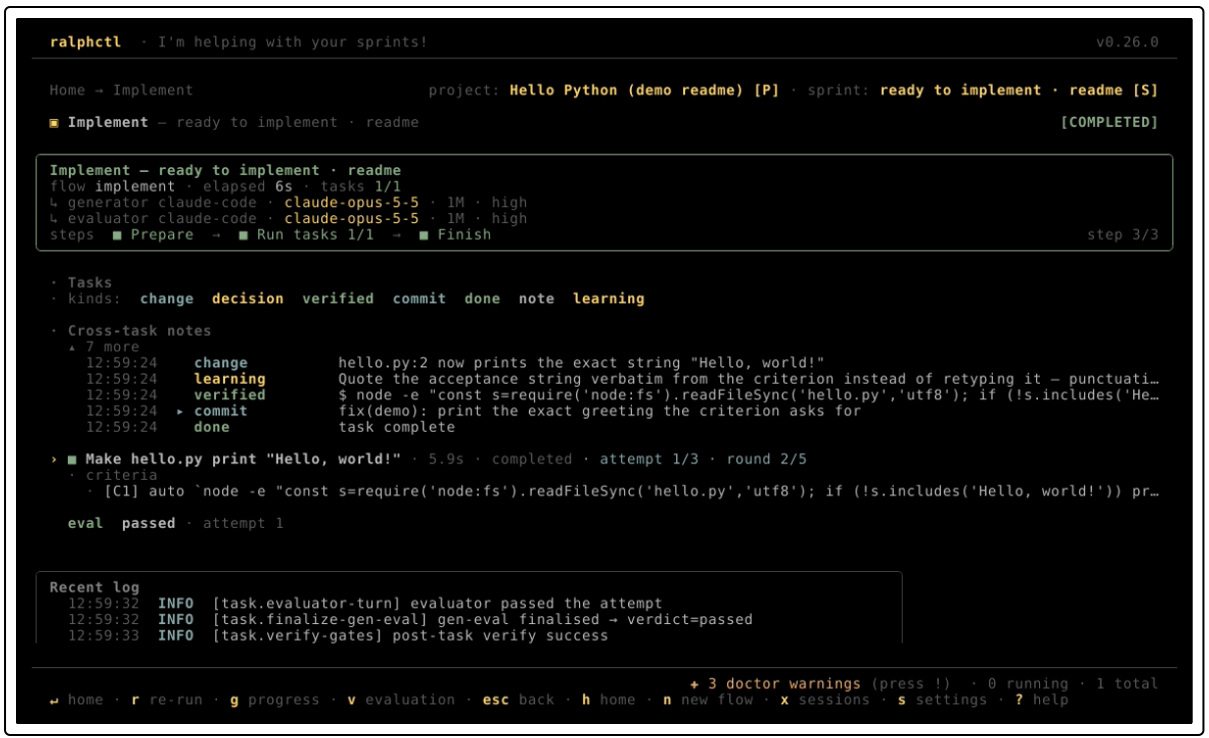
The machine got fast. Did the whole loop? **Over to Dennis.**

Backup · Which model does what?



By default, another lab grades: a GPT model grades what Opus wrote.

Backup · The loop, in the real terminal



1. Round 1: the evaluator says no, and says why.
2. Round 2: the critique applied. Passed.
3. Verify runs again. Then the commit.

ralphctl 0.26 terminal UI.
`ralphctl demo --script` replays a recorded run through the real implement chain: no AI call, no tokens.

`npm i -g ralphctl`
github.com/lukas-grigis/ralphctl · MIT