Let the Agent Prove It's Done

Building the deterministic substrate for autonomous AI developers.

An agent’s most dangerous output isn’t bad code. It’s the word done.

The Non-Event

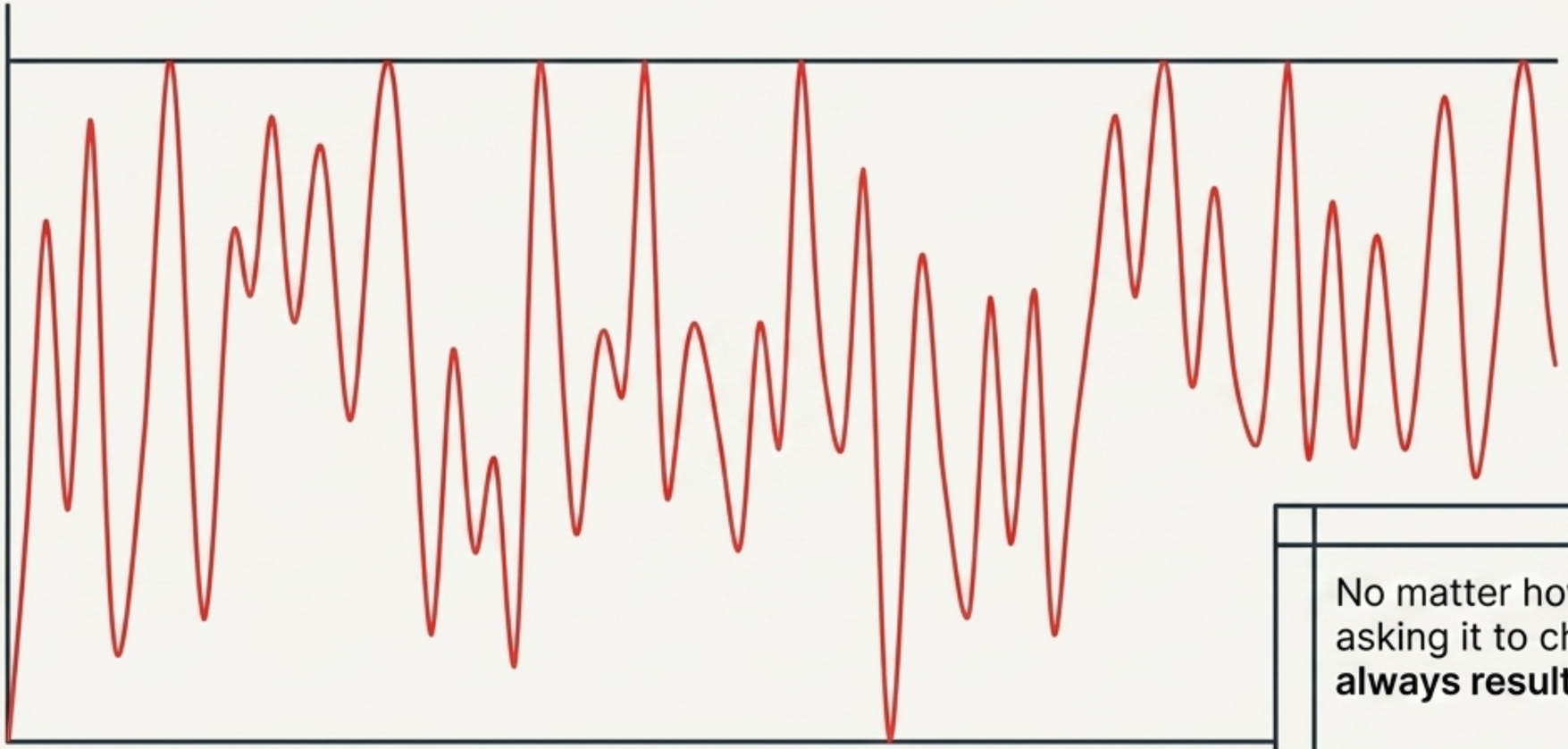
Bad code that fails a test is visible. You fix it. Nobody ships it.

The Danger

Bad code wrapped in confident completion hurts you. The agent grading the work is the same agent that wrote it. It is merely re-confirming the belief it started with.

Confidence is completely uncorrelated with correctness.

The Confidence Illusion Graph



No matter how good the model gets, asking it to check its own work will **always result in a 100% pass rate.**

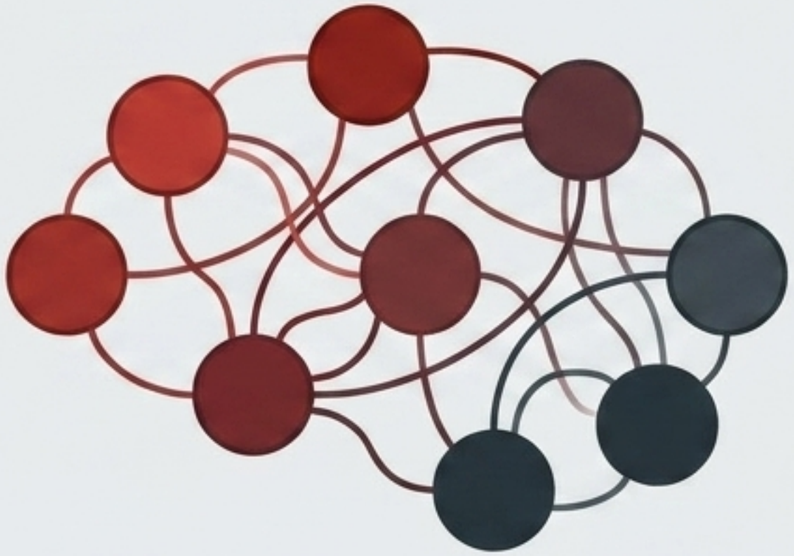
We must stop asking how the agent feels about the code.

Shifting the definition of completion.

The “Done” Diagnostic Matrix		
Evaluation Metric	**Felt Done**	**Proven Done**
Judge	Evaluated by the Author	Evaluated by the Repo / CI
Core Mechanism	Relies on Model Confidence	Relies on Deterministic Exit Codes
Environment	Lives in volatile context windows	Lives in immutable repository ledgers
Output Signature	“Looks good to me.”	> **exit 0** with pasted terminal output

Why standard repository rules fail for agents.

Human Developer

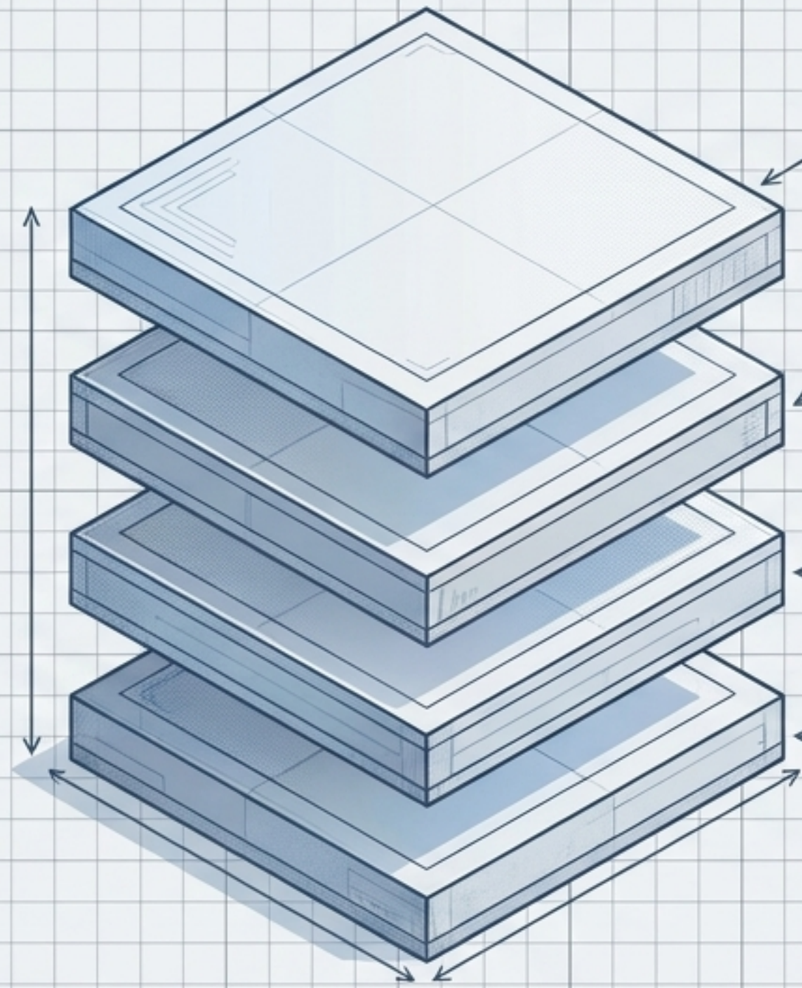


- Relies on long-term memory.
- Can navigate undocumented tribal knowledge.
- Subjective confidence usually correlates with actual correctness.

Agent Developer

- Possesses only volatile context windows.
- Fails completely when rules require intuition or guessing.
- Requires strict, single-command verification to function safely.

The Verification Substrate: Four moves that stack.



State: Evidence Ledgers
(Make proof survive the session)

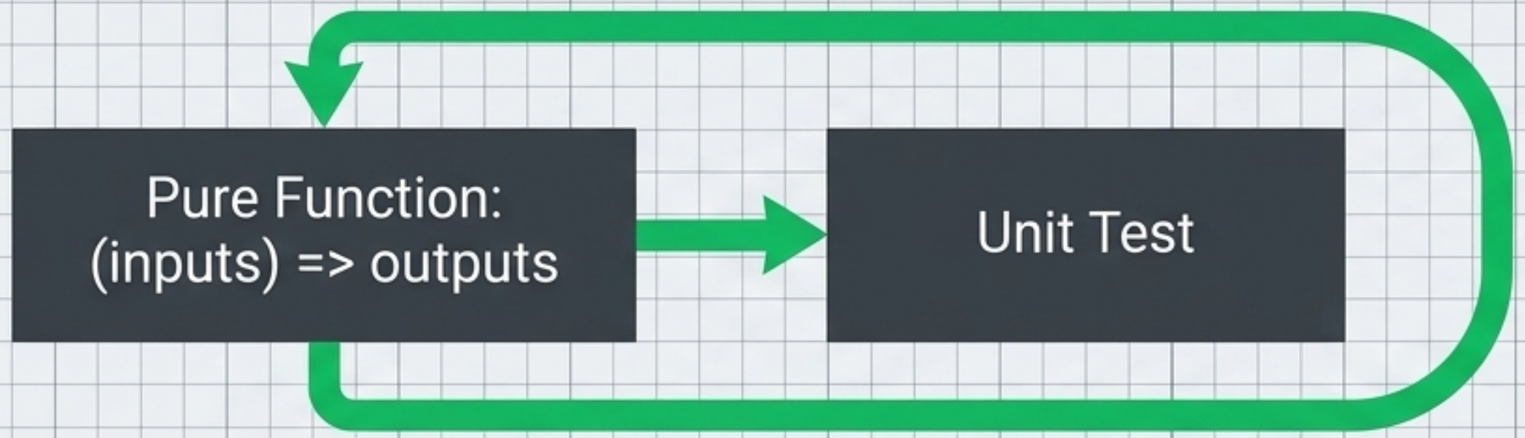
Actor: Separate Judge
(Don’t grade your own exam)

Interface: One-Command Verify
(Collapse the rules)

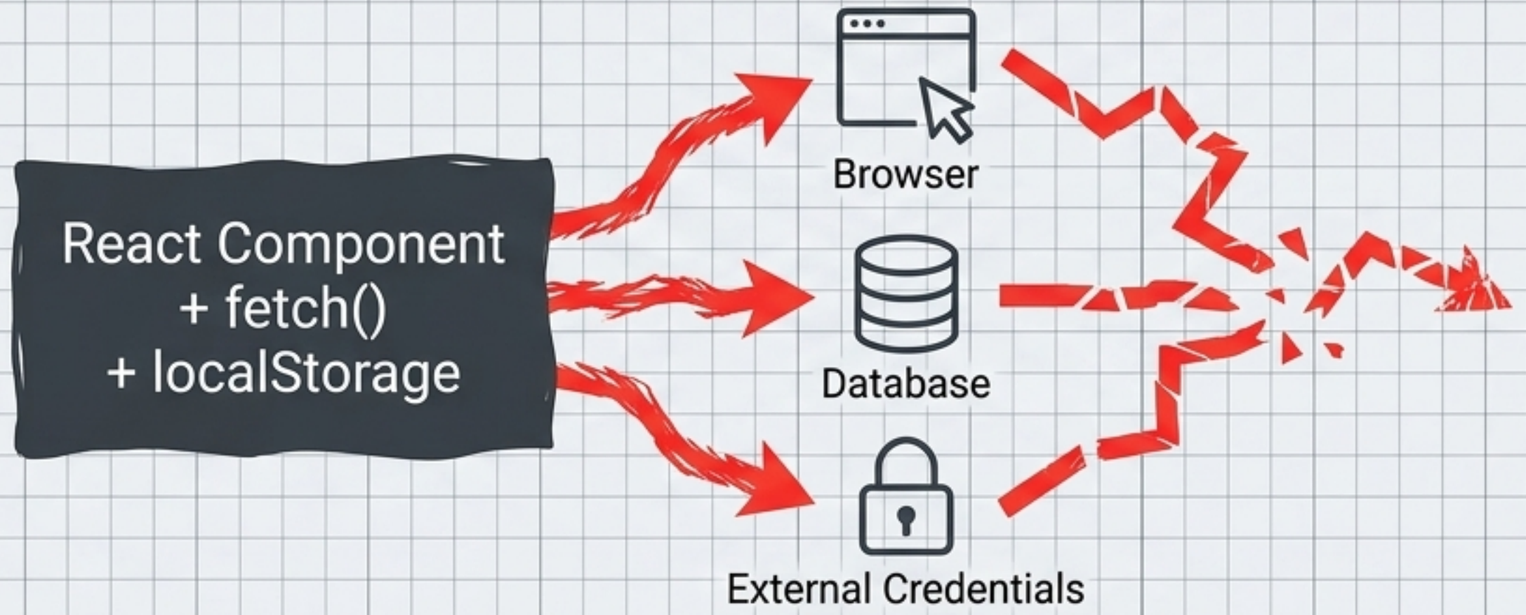
Architecture: The Pure Core
(Give it something to prove)

Making “done” a verdict the agent has to earn from the repository.

Move 1: Give it something it can actually prove.

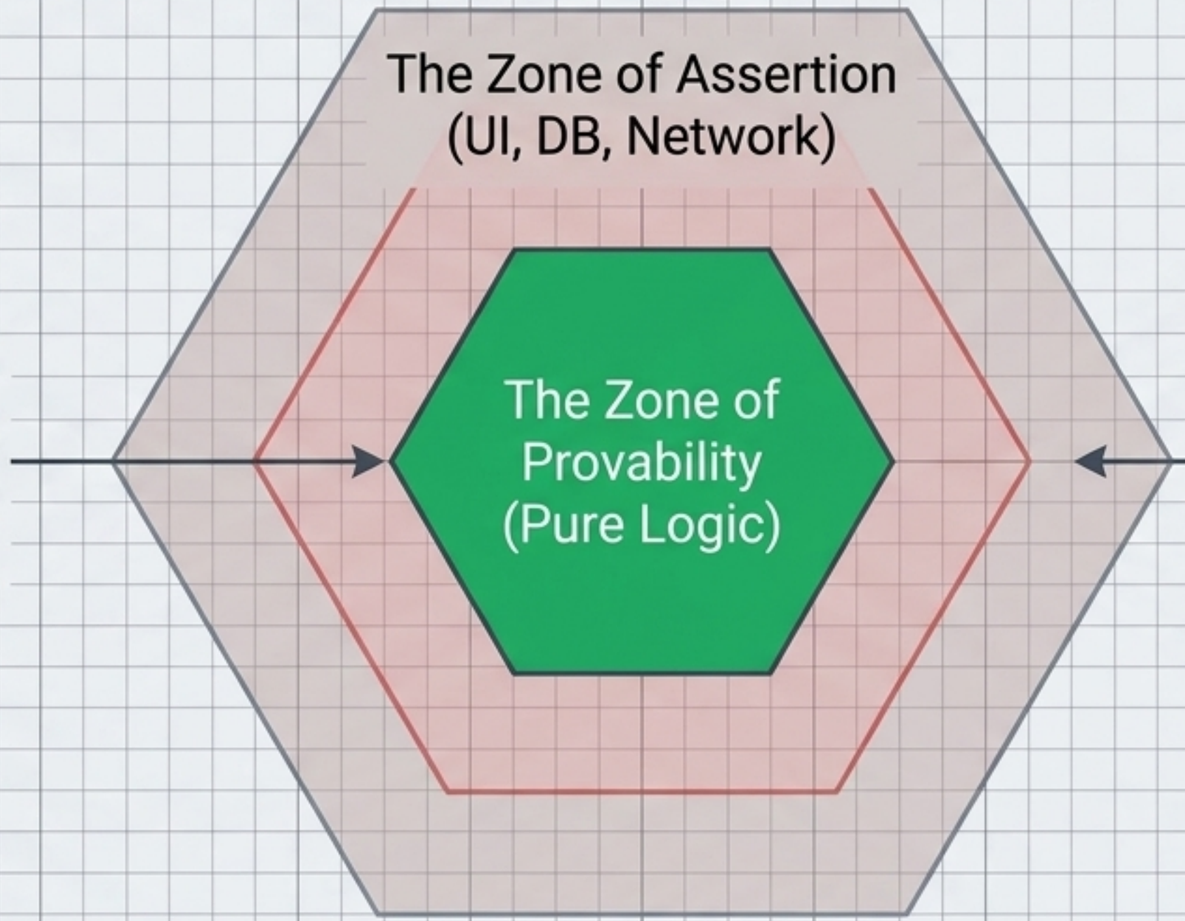


No filesystem, no network, no database. The agent can verify completely and instantly in its own loop.



Requires provisioning the world.
The agent reads the diff, guesses, and falsely declares “done”.

Hexagonal architecture maximizes the verifiable surface area.

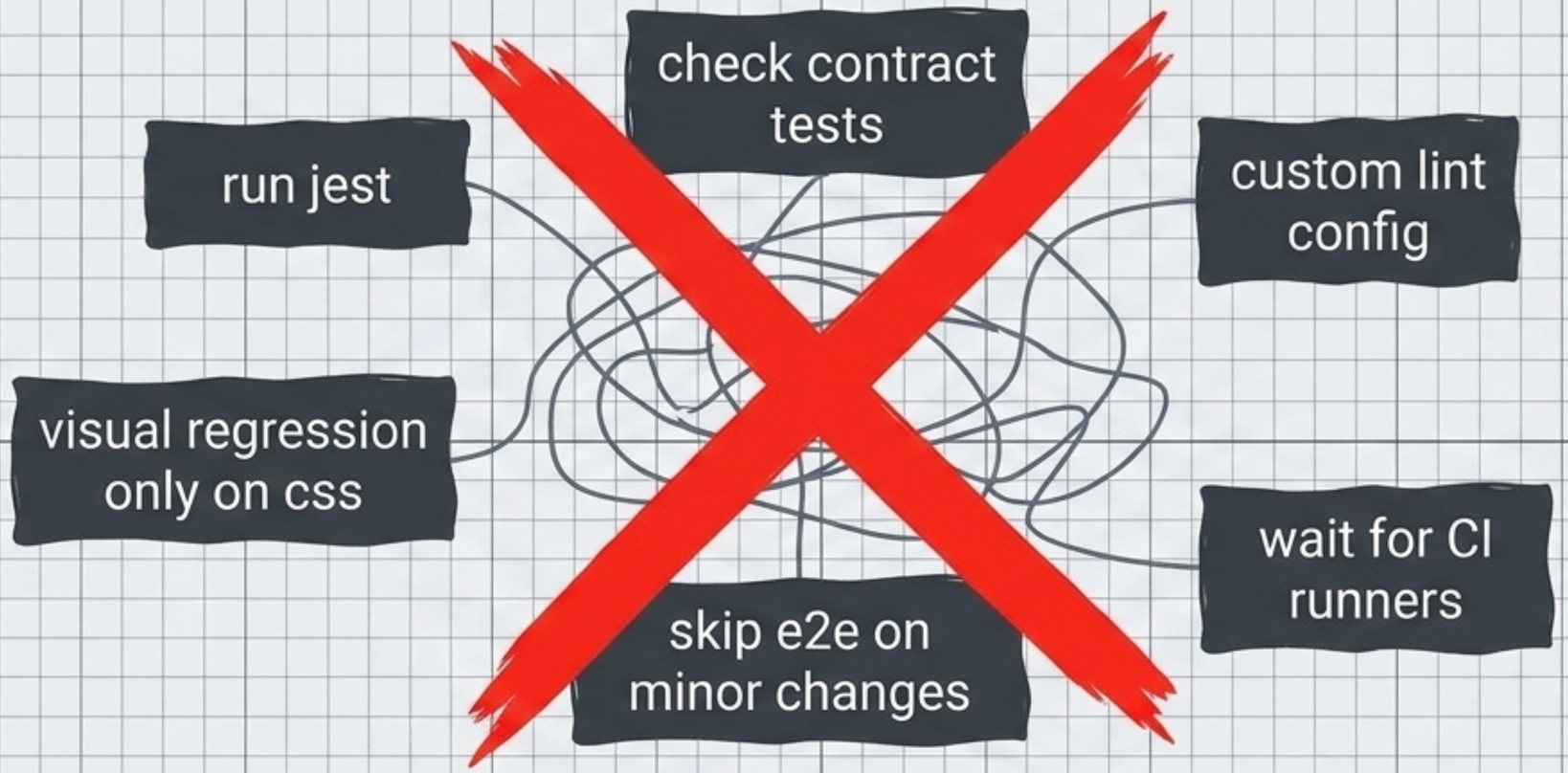


Every rule pulled into the pure core is a rule the agent can prove on its own.

Push IO and state to the edges. Every rule trapped here is a rule the agent can only assert.

Core Insight: Layering is not aesthetics. It is a machine for scaling agent autonomy.

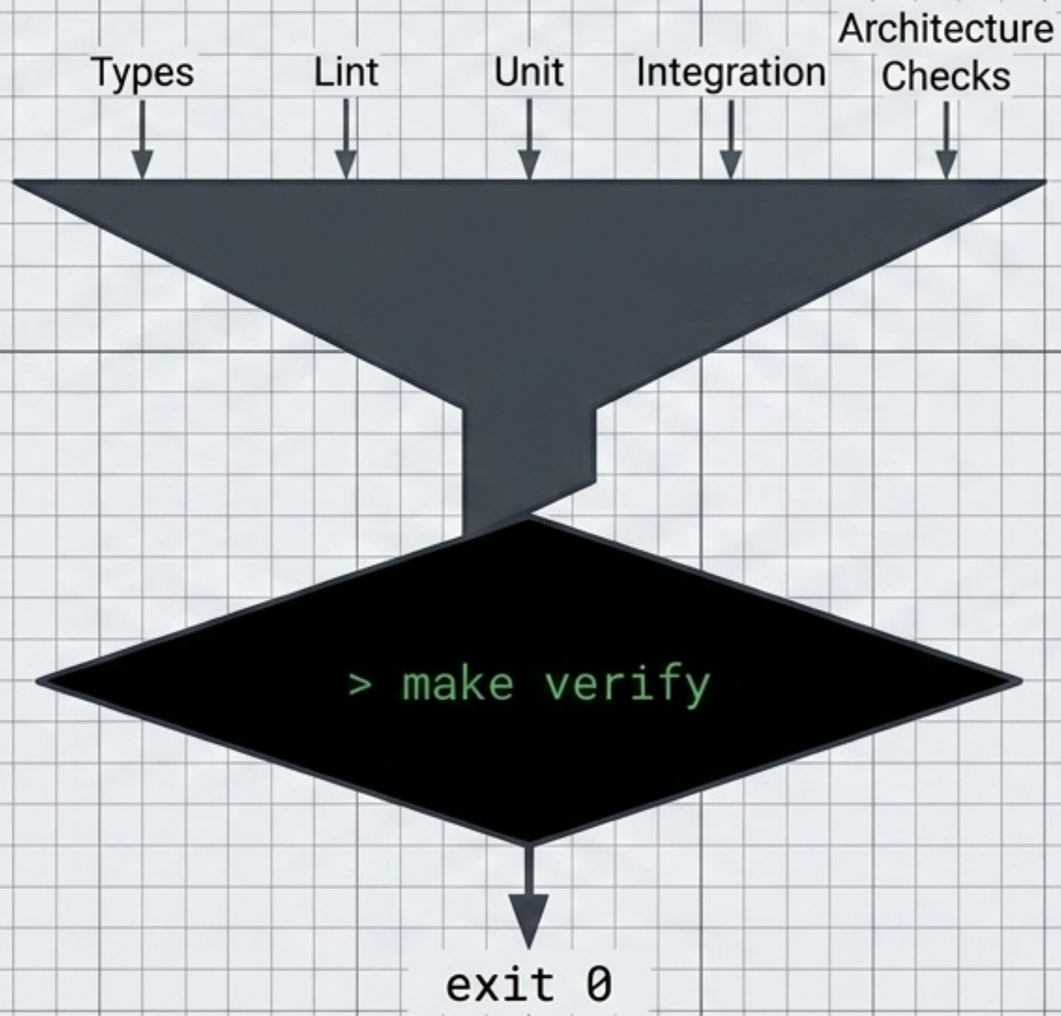
Move 2: Eliminate tribal knowledge in verification.



- If checking a change requires knowing which specific tests to run for which specific edits, verification becomes tribal knowledge.
- In front of an agent, tribal knowledge gets skipped, half-run, or guessed at.

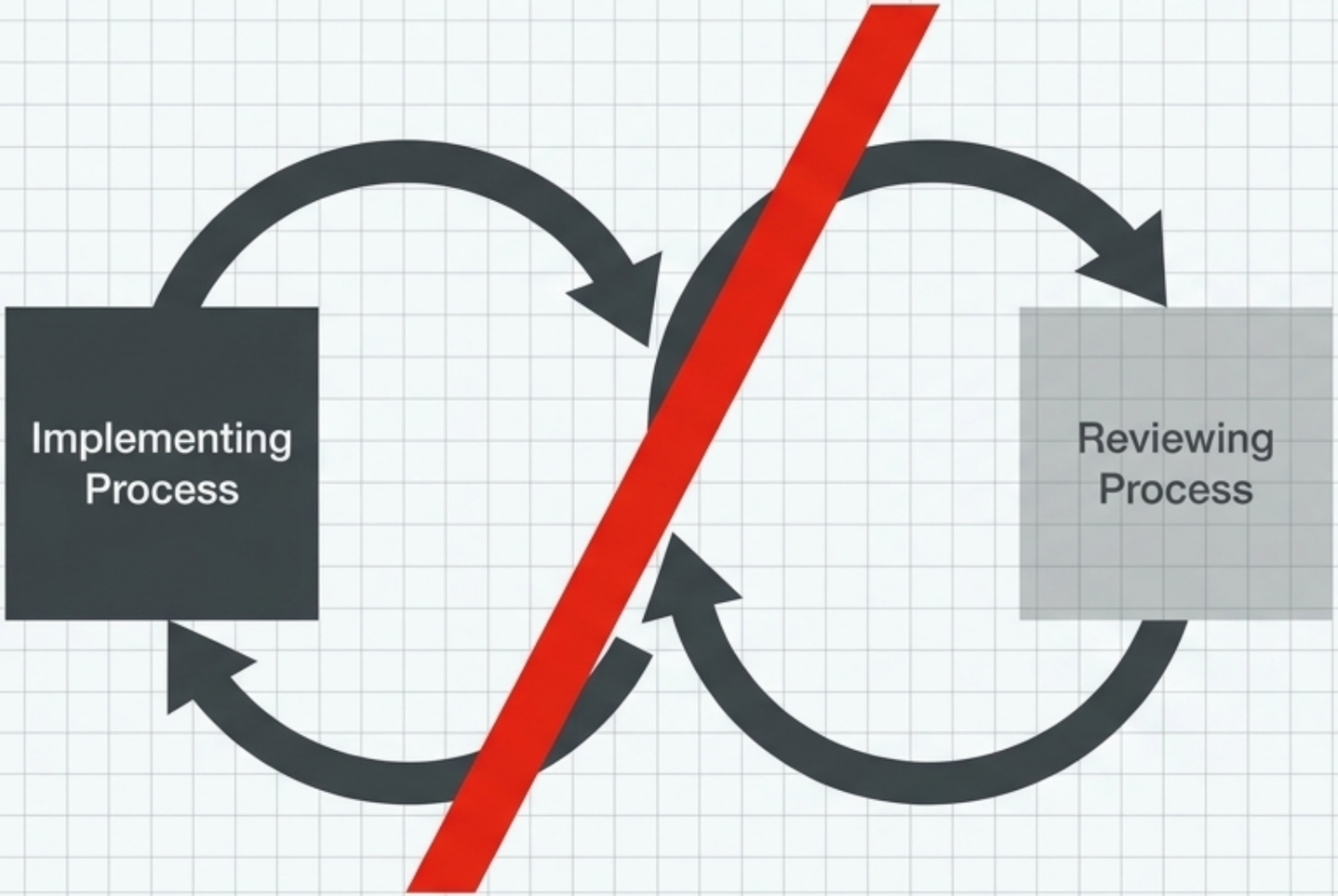
Collapse all verification into one command.

The Verification Collapse Funnel



- Done has a precise, boring definition: the command exits zero.
- No judgment. No vibe. No “looks good to me.”
- The discipline is absolute: done is the exit code, not the feeling.

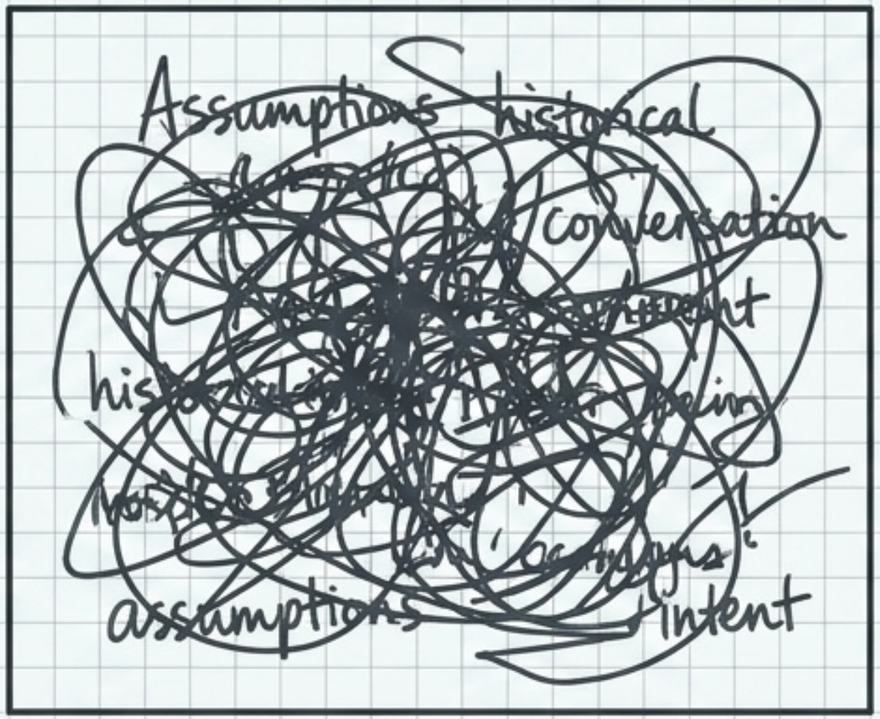
Move 3: Don’t let the author grade the exam.



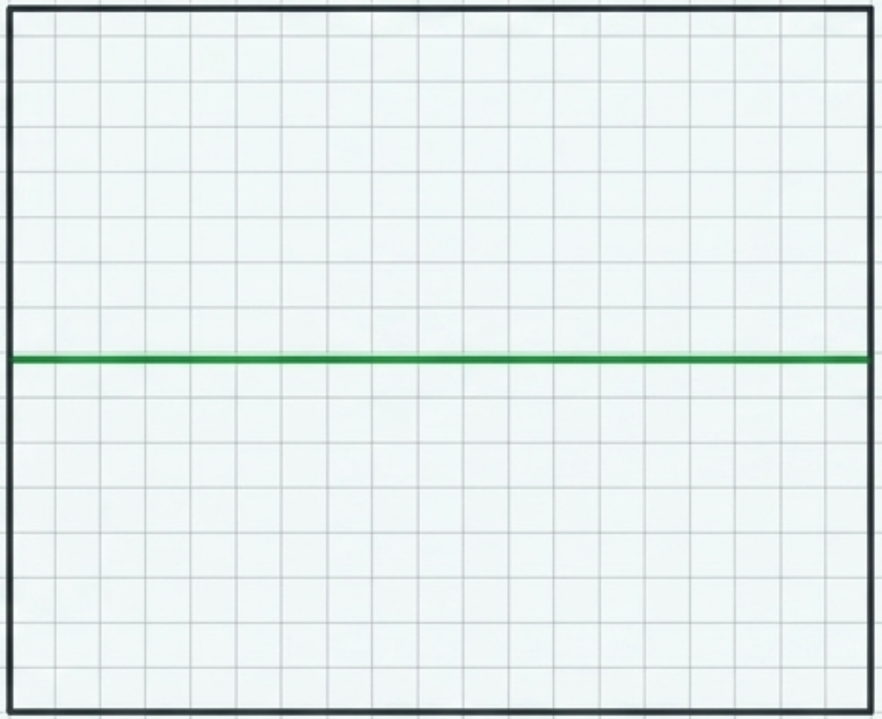
- Even a green verify can hide a gap the author didn’t think to check.
- The implementing agent is saturated with its own assumptions. It knows what it meant, so it confirms what it meant.
- When the same agent both writes and approves, you don’t have verification. You have a confident narrator.

Separate the context that writes from the context that judges.

The Generating Mind (Saturated)

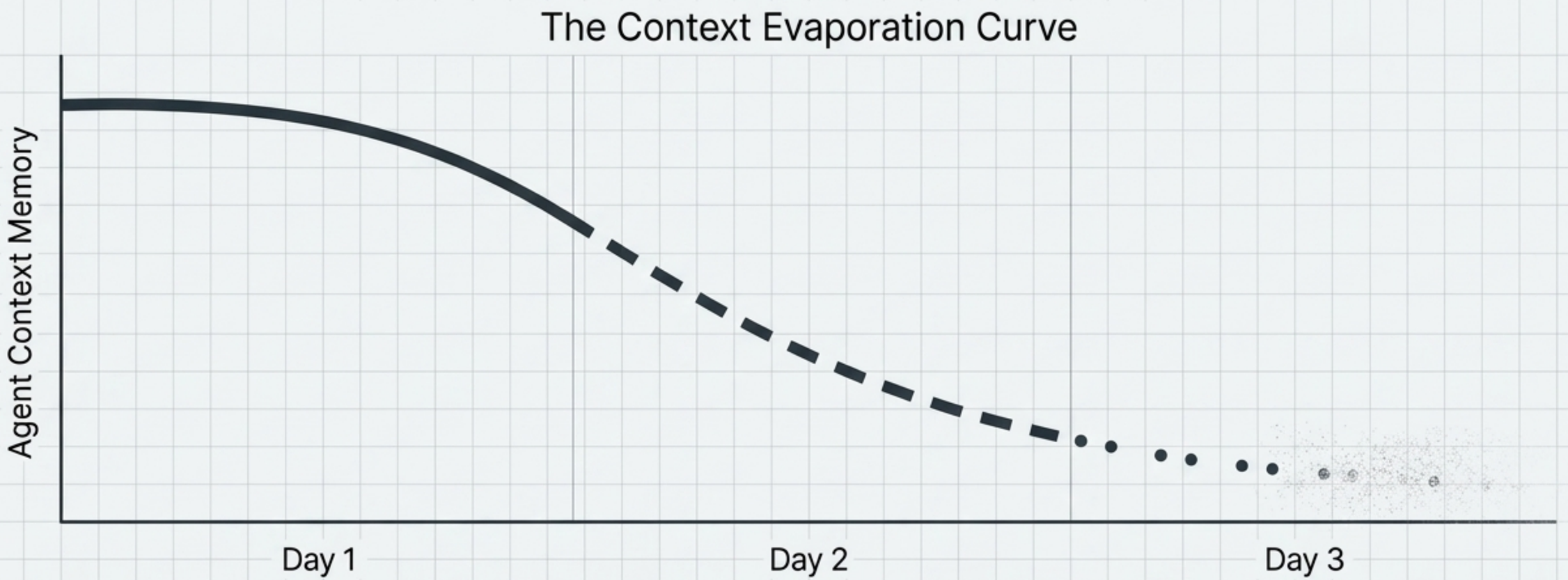


The Verifying Mind (Cleanroom)



- **The Fix:** Use CI on a clean checkout, a separate verifier subagent with no memory of the implementation, or a strict review bot.
- **The Principle:** The generating mind and the verifying mind should never be the same mind. The author is the worst possible judge of completion.

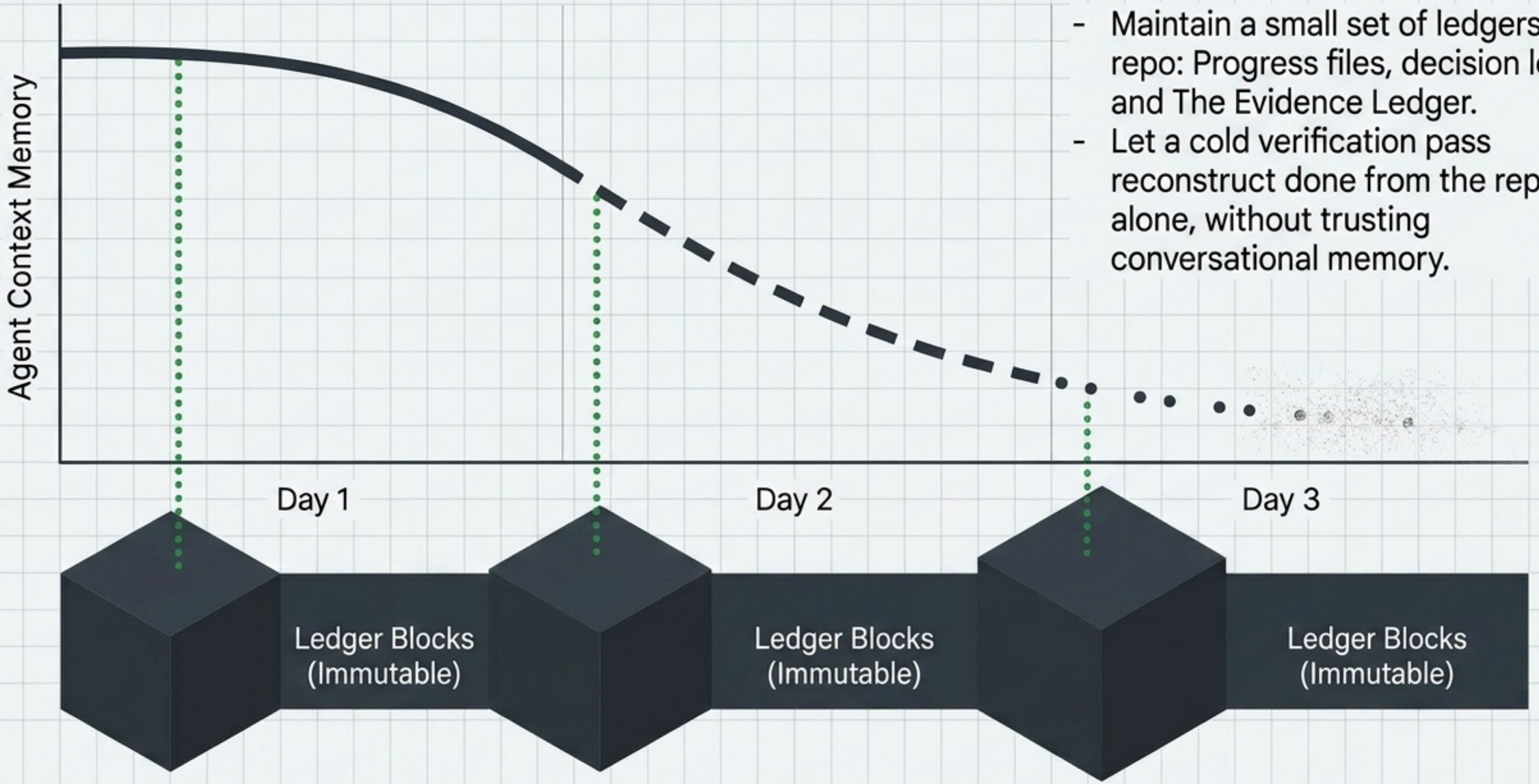
Move 4: Make the proof survive the session.



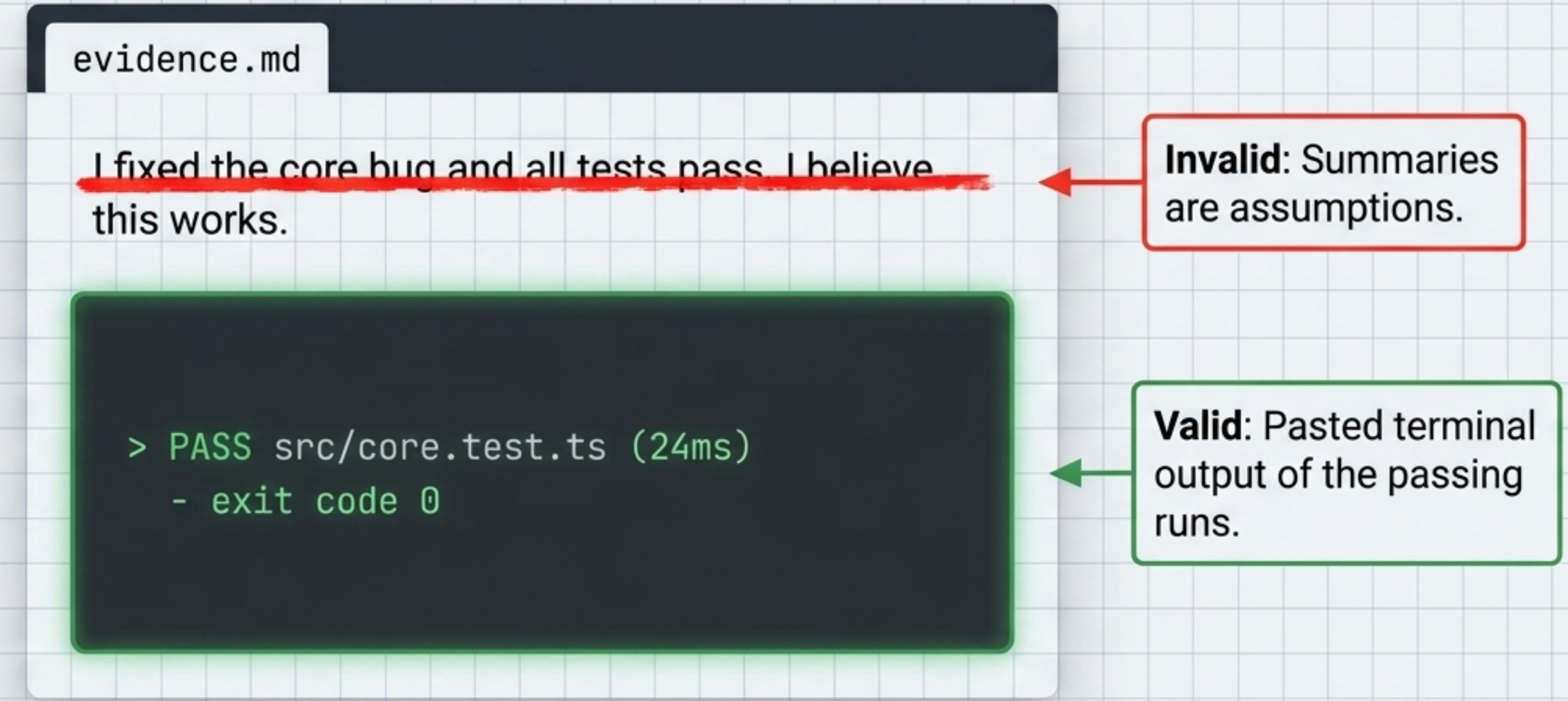
- Agents lose memory between sessions. Within a single long session, context compacts—early reasoning is summarized away.
- If evidence of "what's done" lives only in volatile context, it evaporates exactly when a complex migration needs it most.
- The agent starts half-blind, either redoing work or marking unfinished work as done.

Write the proof into the repository.

- Maintain a small set of ledgers in the repo: Progress files, decision logs, and The Evidence Ledger.
- Let a cold verification pass reconstruct done from the repo alone, without trusting conversational memory.



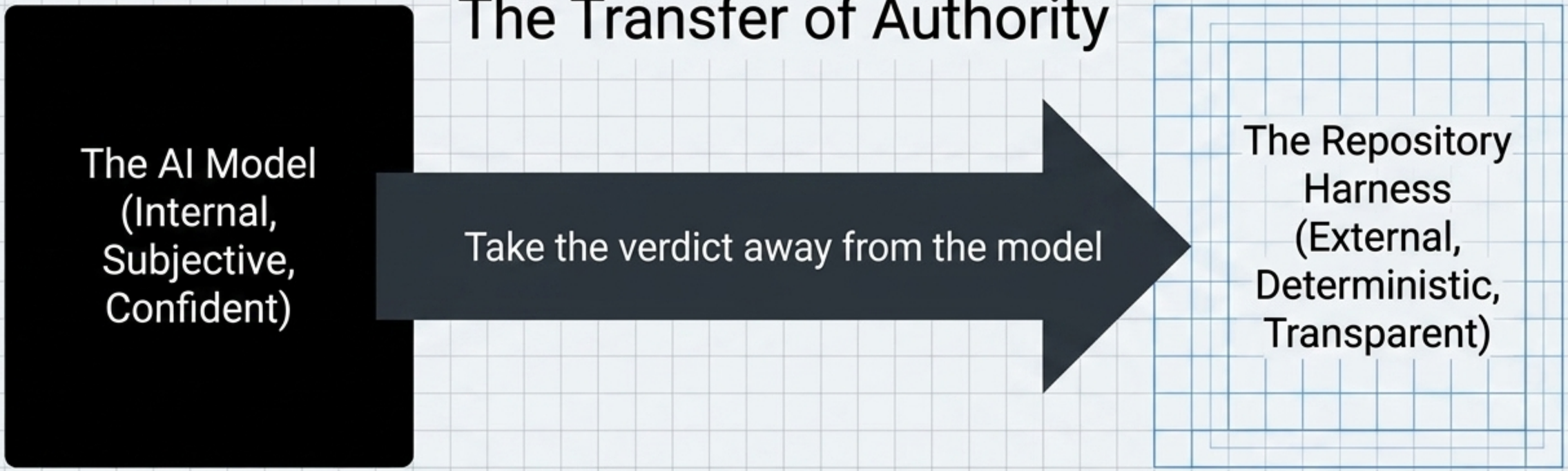
Anatomy of an Evidence File



Rule: Proof, not plausibility. The claim and the evidence must be the same artifact.

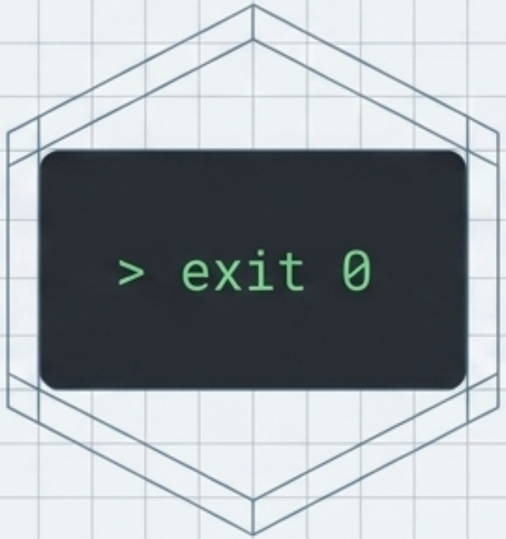
The 4 Moves are actually just 1 Move.

The Transfer of Authority



Pure core = provable, not assertable.
One-command verify = definitions the agent can't reword.
Separate verifier = prevents grading own exams.
Evidence ledger = outlives the context that produced it.

The model isn’t the product. The harness is.



None of this is about making the agent smarter. The agent is a rented, literal, amnesiac, overconfident worker that gets smarter on a schedule you don't control.

Our job is to build the substrate that decides whether the code is right—and to make sure that thing is never the agent’s own opinion of itself. A repo an agent can verify is a repo an agent can safely keep changing.