Making the Right Rule Hard to Ignore

How I made an AI agent more reliable by controlling what it knows, and when

ERIC COHEN · CASE STUDY · STATUS: SHIPPED

My agent turns raw financial data into analysis an investor acts on with real money. It runs on rules I wrote: build a peer group from the rivals a company names in its own SEC filings, pick metrics that compare honestly across very different businesses, and stop to hand the real judgment calls to the investor. Rules govern more than the analysis. A product playbook loads before it works its own backlog, ranking the work with RICE, MoSCoW, and WSJF. Writing the rules was the easy part. **Getting the agent to act on the right one at the right moment was the hard part.**

I solved it by changing what the agent holds, and when. **That one change cut the context the agent carries into every run by more than half, and made it more reliable, not less.**

> *An agent can ignore any rule it holds. My job is to make the one that matters the hardest to ignore.*

THE DANGER

A skipped rule looks exactly like a followed one.

Two of the calls my agent makes belong to the investor: which companies are true peers, and which metrics rank them. The yardstick that fits a software company misreads a bank. Neither has one right answer, and being wrong on either moves real money.

A skipped rule leaves no mark. The analysis comes back finished, a wrong peer set reading as clean as a right one, and the agent has made a call the investor was meant to make. **Verifying the math catches a broken formula. It cannot catch a sound formula built on the wrong peers.**

THE FIX

The more rules I handed the agent at once, the worse it followed them.

A single rulebook, loaded every session, was the right call when the system was small. Then it grew, and the more rules that arrived up front, the harder the one that mattered on a given run was to find under a hundred that did not apply. Cost was not the real problem; prompt caching keeps a big standing context cheap to carry. **Attention was.** Every rule in the window competes for the model's focus, and the ones that don't apply right now are just noise.

So I split the rulebook. A thin core stays loaded for every run, and everything else moved into playbooks that load only when the work calls for them. Deciding what goes where was the hard part, and my first cut was wrong: I grouped them by topic, and one catch-all file ended up answering to three unrelated triggers. The rule I settled on is sharper. **Split by the trigger that pulls a playbook in, not by subject;** if two rules don't load on the same event, they don't share a file. The mapping from an action to the rule that governs it is known and exact, so I match it on the trigger rather than letting the agent fetch rules by similarity to the task. A known mapping beats a guess.

Loading on demand only works if the load happens, so the gate is hard: the agent cannot start a data run or a commit until the playbook for that action has loaded. **An instruction to load it first is a request the agent can miss. A gate is not.** The split cut the context the agent carries into a run by 56%. The rule that matters is no longer buried, so the agent is far less likely to skip it. Extending the eval log to measure rule adherence directly could let me track that improvement over time.

WHAT CHANGED

The split that made the agent lean is the same one that made it more reliable.

It carries less than half of what it once did, reaches just as far, and follows the rule that matters more closely. It costs less to run, too.

The approach reaches past finance. Any agent you have to trust runs on rules, and loading all of them at once buries the one that matters. Writing good rules is the easy part. **Getting the right one in front of the agent at the right moment, with less in its way, is what earns trust, lifts performance, and lowers cost.** That is the harder part, and the one I build.

This case study covers the architecture and the reasoning behind it. The rule files, hook internals, and exact gate logic are intentionally omitted.

Eric Cohen · cohenemc@gmail.com