

Not every "agent" should be an LLM call.

A practical split for building reliable, cost-sane agentic systems.

The default failure mode

Most agent pipelines route every step through the model: validation, formatting, retries, even the final write.

The result: high cost, high latency, and non-determinism in exactly the steps that should never vary.

Wrong question vs. right question

"Should this be agentic?"

is the wrong question.

The right question:

Which step needs judgment, and which step just needs correct, repeatable execution?

The test I use

If you could write a test case for the "correct" output:

-> Python function

deterministic, cents per run

If "correct" is a judgment call, not a spec:

-> LLM sub-agent

judgment, dollars per run

Thin harness vs. fat harness

Thin harness

The model designs the pipeline once. Code runs it every time after.
Cents per run.

Fat harness

The model replans the whole task on every run. Dollars per run.

Who uses which, and why

Fat harness (on purpose):

Claude Code, Cursor. The task really is different every time -- novelty is the point.

Thin harness (in production):

Most shipping SaaS AI features. The task repeats, and a wrong answer compounds.

Here's the proof

70%

of the time, on short tasks, top models get it right

23%

of the time, on longer, messier tasks -- same models

The longer and more open-ended the task, the less you can trust the model alone.

Let code handle the 80% that's just repeatable steps.

Save the model for the 20% that actually needs judgment.

Look at the last agent you shipped.

Which step was a pure LLM call that a deterministic function could've handled just as well?

Jashan Sadioura

Follow for more on agent architecture and AI product design