

AI ENGINEERING · PART I · 14-SLIDE SUMMARY

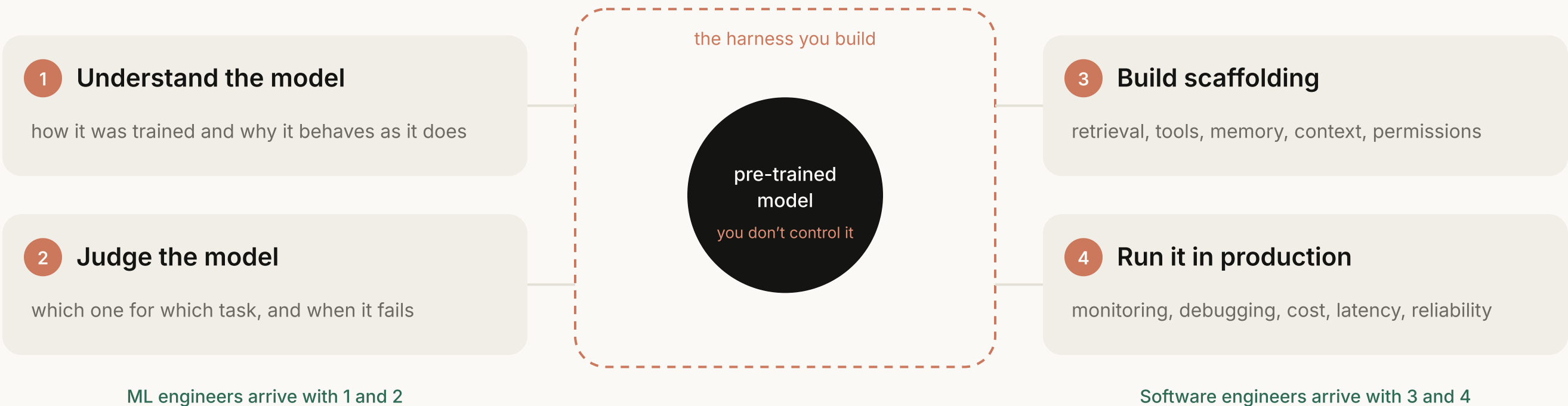
The art of AI engineering — Part I

Building reliable products on top of pre-trained models: knowing when to hire the brain, and evaluating it honestly.

From the deployment decision to reliable evaluation

PART 1 · SLIDE 3 The role 03 Scope and core skills	PART 2 · SLIDES 4-8 Deployment decision 04 Why the decision is hard 05 Prioritisation criteria 06 Decision rule and sequencing 07 Case study: software engineering 08 Case study: venture capital	PART 3 · SLIDES 9-13 Evaluation 09 Why AI fails silently 10 Model selection: benchmarks 11 Model selection: agent evals 12 Outlook: evals as engineering 13 Outlook: system-level evals
--	---	---

Build reliable products around a brain you don't control.



It is non-deterministic, and a wrong answer sounds as persuasive as a right one. **The hardest part isn't building; it is being confident that what you built works.**

Why the first judgement call is hard.

1

An unreliable new part

a workflow that already runs

faster, cheaper, happier customers

broken trust, ballooning cost

The business already runs. AI can speed it up, or break the trust it runs on.

2

The ground keeps moving

TASKS A MODEL CAN FINISH (METR)

16 h+ · 2026

~4 min · Mar 2024

Models last 6 weeks to 12 months. Build like seismic design: **the least scaffolding that does the job.**

3

Two tracks of work at once

CLASSIC TRACK

fix bugs · ship the features customers ask for

AI TRACK

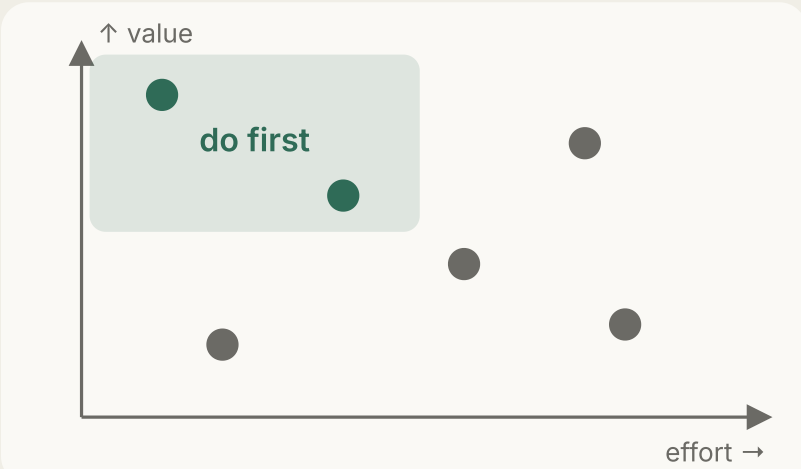
a new model every few months to fold into your product

Everything classic software needs, plus a second stream: at least double the work, on the same team.

Score every task you could automate on three things.

EFFORT VS REWARD

Value for the effort



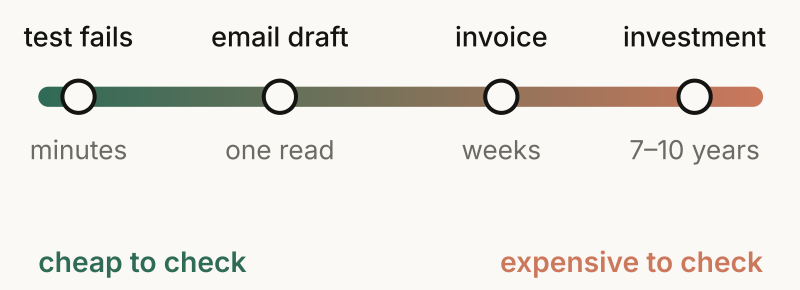
The least technical effort for the highest commercial value.

The diagram shows a scatter plot with 'value' on the vertical axis and 'effort' on the horizontal axis. There are six data points represented by dots. A light green rectangular box labeled 'do first' encloses the two points that have the highest value for the lowest effort. The other four points are scattered at higher effort levels for varying values.

VERIFICATION COST

How cheap is checking?

HOW FAST YOU LEARN IT WAS WRONG



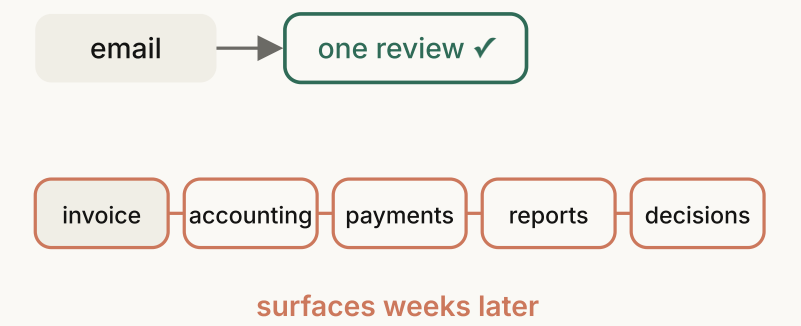
When the brain is wrong here, how quickly and cheaply will you know?

The diagram is a horizontal timeline with four stages: 'test fails', 'email draft', 'invoice', and 'investment'. Below these stages are time markers: 'minutes', 'one read', 'weeks', and '7-10 years'. A green bar under 'test fails' is labeled 'cheap to check', and a red bar under 'investment' is labeled 'expensive to check'. The timeline shows a progression from quick and cheap verification to slow and expensive verification.

BLAST RADIUS

How far does it travel?

HOW FAR A MISTAKE TRAVELS

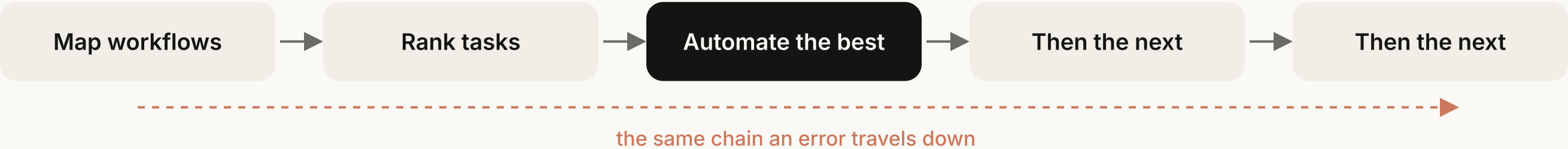


A draft email stops at one review. A wrong invoice amount flows on for weeks.

The diagram shows a flowchart of a process flow. It starts with 'email', which leads to 'one review ✓'. Below this, a sequence of boxes represents the flow of information: 'invoice', 'accounting', 'payments', 'reports', and 'decisions'. A red arrow points from 'invoice' to 'accounting', and another red arrow points from 'reports' to 'decisions'. The text 'surfaces weeks later' is written below the 'reports' and 'decisions' boxes, indicating the delay in catching a mistake.

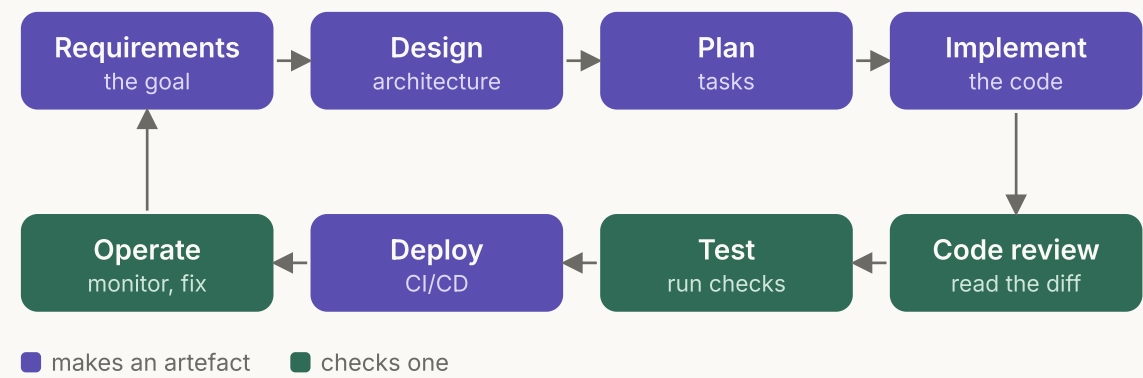
Hire the artificial brain where you can afford to check its work — and have a proper plan for if and when it fails.

Map your organisation's workflows, processes, software and their dependencies first.



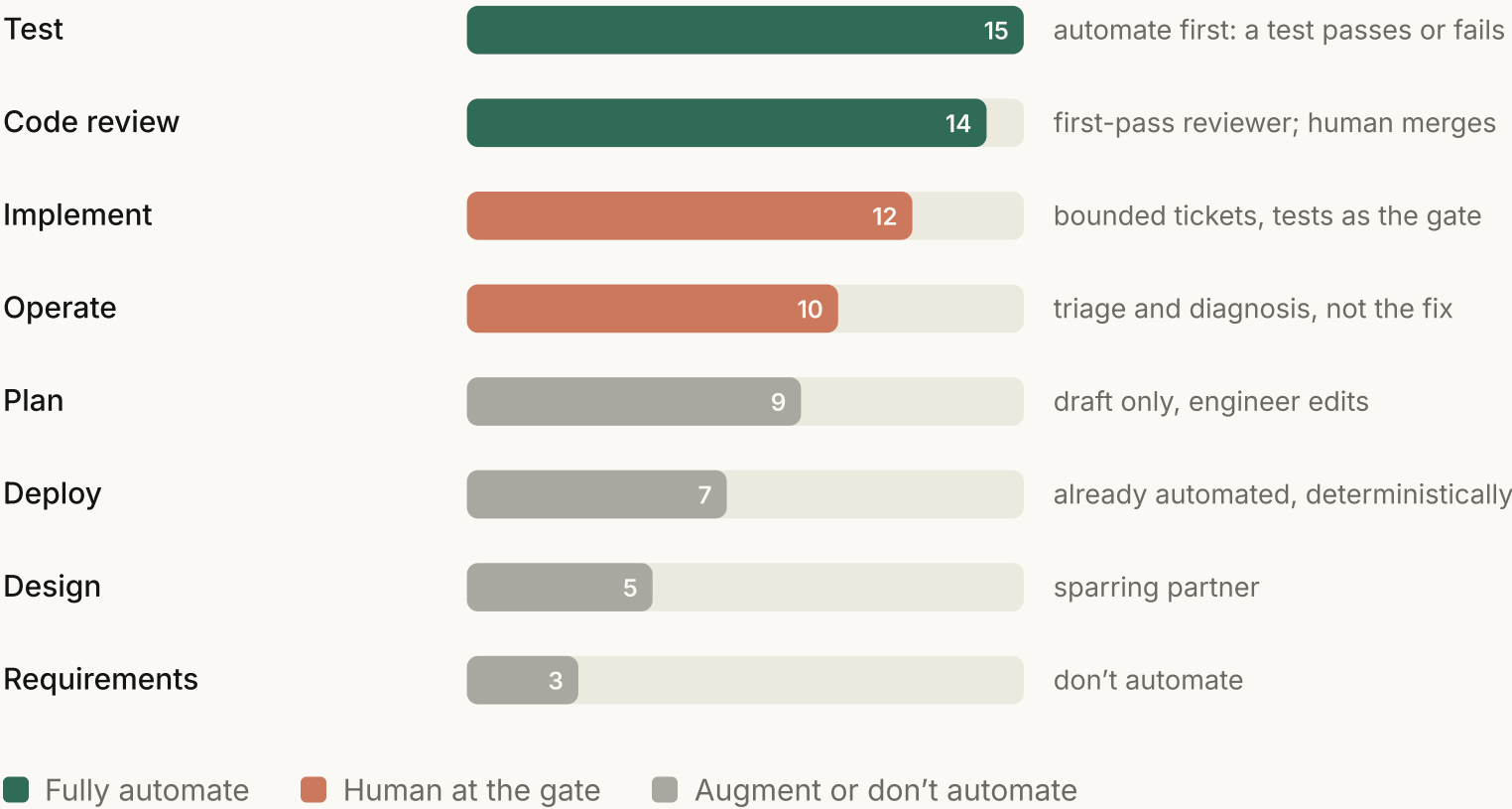
Design rule: automate first where a mistake is stopped in the fewest steps, and put a human in the loop wherever it would travel.

Software goes first: checking is nearly free.



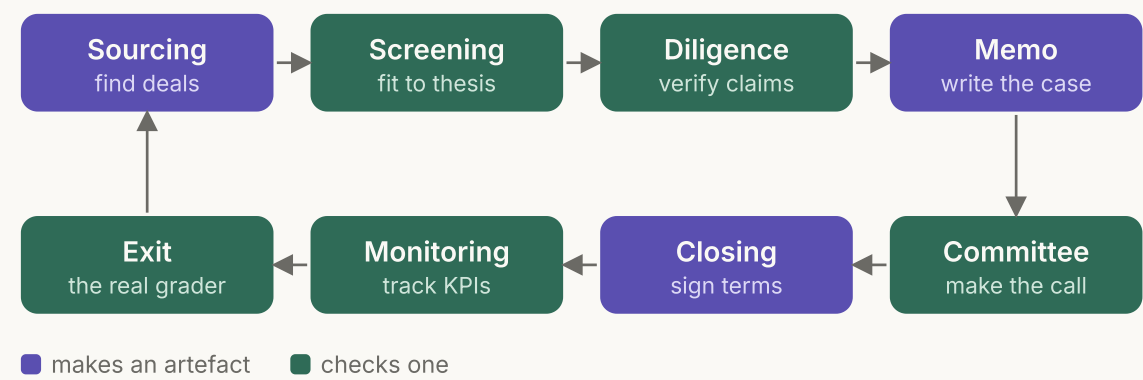
The check stages are the cheapest to automate: **something cheaper than the model already verifies them.**

SCORE OUT OF 15 · EFFORT VS REWARD, VERIFICATION COST, BLAST RADIUS



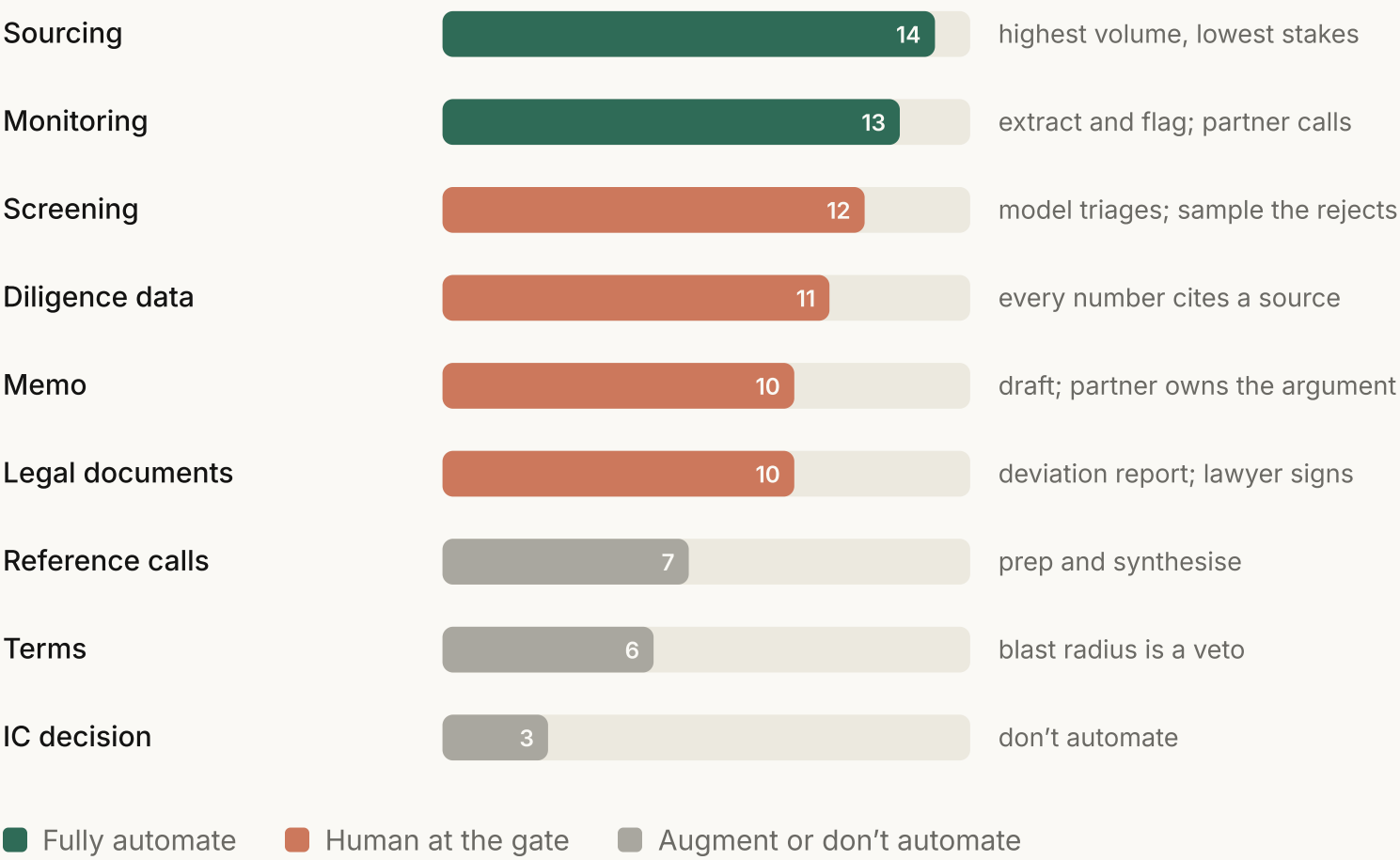
Skip the guardrails and you haven't automated the stage. **You have moved its verification cost onto your customers.**

VC splits in half: automate volume, keep the call human.



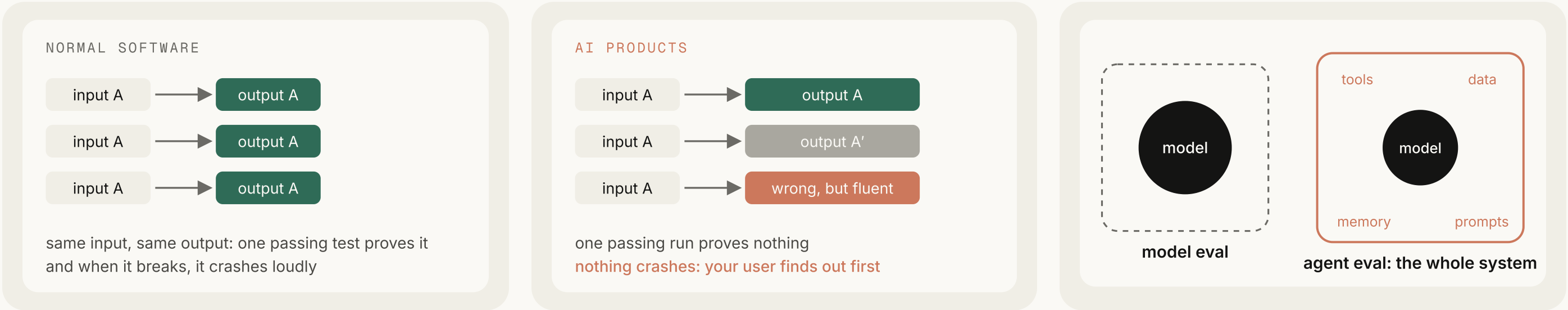
About 20 decisions a year, and the truth arrives 7–10 years later. In diligence, “done” means refusal: the best model says “not in the data room” every time it should.

SCORE OUT OF 15 · EFFORT VS REWARD, VERIFICATION COST, BLAST RADIUS



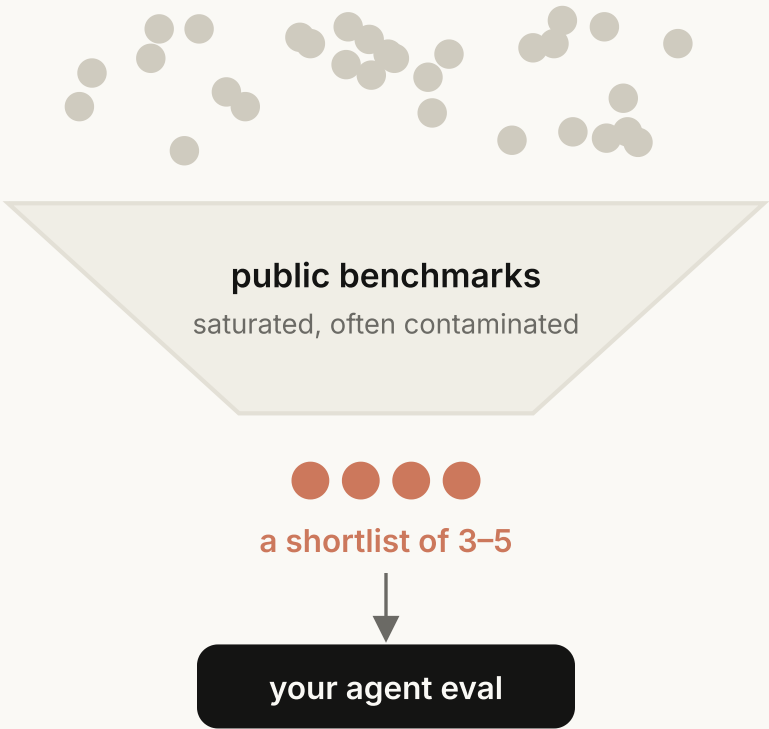
The hardest part of AI engineering: it fails silently.

An eval is a repeatable measurement of whether a model or agent is good enough at a specific task to be trusted with it.



Grading has grown harder and dearer in every era: from string matching, to answer keys, to human votes and LLM judges, to domain experts marking real work. The essay has the full history.

Public benchmarks are a scouting tool, not a verdict.



Even an inflated score still carries **three signals**:

SIGNAL 1

Exposure

A model trained on that material has seen that kind of knowledge.

SIGNAL 2

Intent

A lab that optimised for the benchmark tells you which domain it wants to win.

SIGNAL 3

Ecosystem

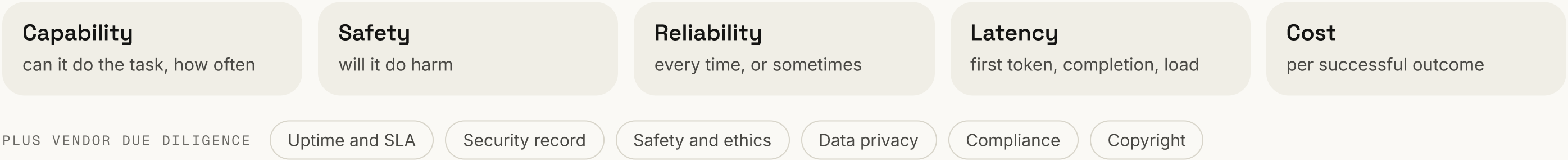
The top model gets the tooling, community and support around it.

Then test it on your own work.



An LLM judge is the best trade on speed and cost, not on accuracy. **It only works if you calibrate it**, and a false pass is worse than a false fail.

FIVE CRITERIA TO TEST AGAINST



Evals become software engineering.

1

Evals are code

eval suite v1
written ahead of the next model

reviewed in a PR

owned, like unit tests

experts write tasks, engineers own the infra

Versioned, reviewed, owned. When a new model drops, the suite tells you overnight which bets paid off.

2

The test pyramid

end-to-end
full trajectory · nightly

integration
one workflow

unit
one prompt, one tool · every commit

cheap and frequent at the base

Split capability evals (a hill to climb) from regression evals (an alarm if they drop).

3

Report reliability

ONE RUN PASSES

75%

THREE RUNS IN A ROW PASS

42%

pass@k: can it do it at all?
pass^k: does it do it every time?

for customers, the second one counts

A 75% agent passes three runs in a row only 42% of the time.

...and then they run the whole system.

4

The eval becomes the router

```
graph LR; input[input] --> router[router<br/>reads evals]; router -- easy --> small[small]; router -- hard --> frontier[frontier];
```

Pick the cheapest model that clears the bar, per input. The router is a brain for hire too: it needs its own eval.

5

Diff the model

SAME EVAL TASK, TWO MODELS

old

new

score: same

path: changed

diff the trajectories, not just the scores

A flat score with a changed path is a change you haven't understood yet.

6

Evaluate the system

same model

18 of 30 picked the same branch name and colluded on prices within 3 rounds

Agents on the same model think alike. That only shows up with the whole system running.

THE SHORT VERSION

The hardest part isn't building. It is being confident that what you built works. So hire the brain first where **checking is cheap** and the **blast radius is contained** when things go wrong, and build the evals to know **how much you can trust it**.

READ huyentran.tech/writings/art-of-ai-engineering
