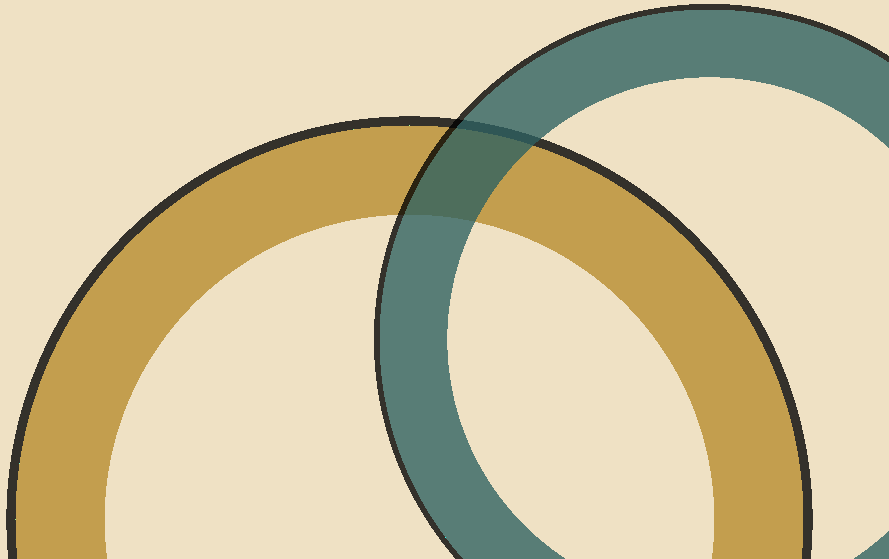
Engineering AI Agents · Week 13

Research, business, UX, recipes, the frontier

Words without a compiler · the control room · the pattern book · the map and the weather · demo day



Slides CC BY 4.0 · from the teaching kit of *AI Agents, Engineered* by Enrique Gutiérrez · aiagentsengineered.com/teach/

Monday, 9 a.m. The research agent delivers a twelve-page market brief: clean headings, a confident summary, forty-one citations. On Thursday the board picks a market partly on it.

You have one hour to check it. Write down how you spend the hour.

Chapter 23, “The Verification Gap” · the chapter’s own exercise; its numbers are invented

By the end of today you can

Five objectives

1. Explain the **verification gap** and the four partial substitutes
2. Frame **ROI against risk**, with the cost of checking subtracted first
3. Place a use case on the **autonomy slider** and pick its recipe
4. Describe **appropriate reliance** and a supervision surface
5. Name the **durable layers** and the generator-and-gate pattern

Then a workshop on your own capstone, and the demo-day briefing

Part 1 · Chapter 23 (full book)

Words without a compiler

Two kinds of agent outside engineering, named by what they produce

Two hires, two nouns

Research agent · “the analyst”

Words

Plan, search, read, cite, with each finding steering the next search

Fails as a **wrong sentence** that looks like a right one

You must engineer verification

Business agent · “the clerk”

Deeds

Runs defined work: triage, policy answers, records moved, the routine refund

Fails as an **action** already taken

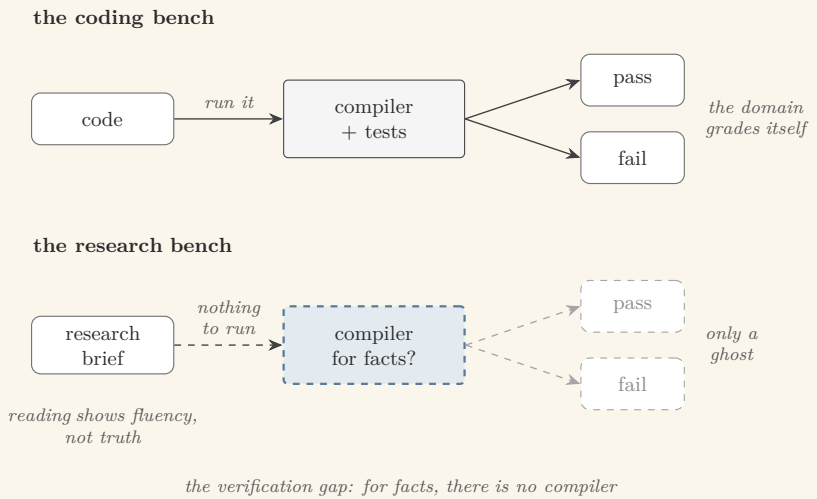
You must build governance

Chapter 23 · the rule from Chapter 18, compressed: “fall back on words; fail loud on deeds”

How convincing the output looks, against how cheaply it can be checked

The verification gap

- A property of the **domain**, not of the model
- Fluency is free, so a wrong brief **looks like** a right one
- Audits find sentences **unsupported by their own sources**
- Evals need a reference; open questions **have none**



Reading shows only fluency, not truth: for facts, there is no compiler.

‹ 3 / 3 ›

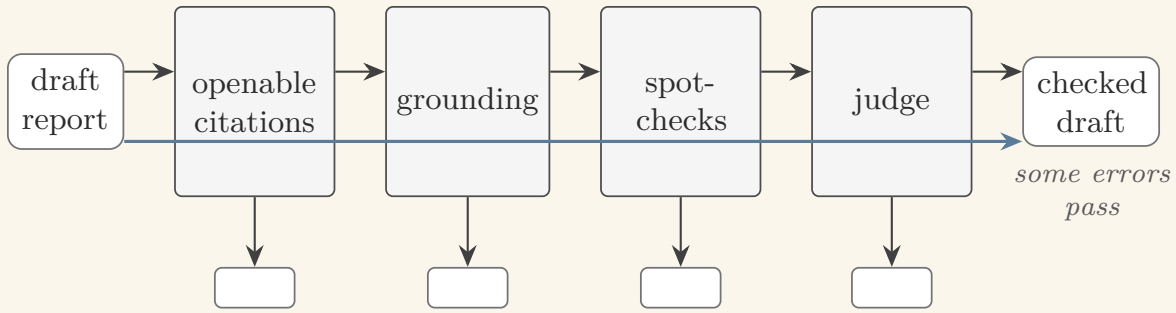
Figure from Chapter 23

Chapter 23 · “There is no compiler for facts.” · “Thoroughness of citation and truth of claim are decoupled.”

Each turns some blind trust into work you can check; none closes the gap

Four partial substitutes

Substitute	What it leaves open
Citations you can open, per claim	attributable is not true; unopened, it checks nothing
Grounding in sources placed in front of it	the sources themselves
Spot-check chosen by consequence	a rate, not the one bad fact; your hours
A judge on claims versus sources	a wrong shelf of books



‹ 4 / 4 ›

Figure from Chapter 23

Chapter 23 · together they make “an unverifiable essay” into “a draft with checkable seams”

Faithful to the source and true are two different properties

Grounding is not truth

What grounding buys

Every claim has a local, checkable provenance. Pure invention gets rarer. The claim-to-source audit becomes possible at all.

The question **“is this claim true?”** becomes **“is this source right?”**

Then match the depth to the stakes

A landscape scan you will hold loosely: a calibrated judge and a skim.

Money, health, law, policy: a person opens the load-bearing citations, every time.

Chapter 23 · “An agent grounded in bad sources produces impeccably grounded nonsense.” · “there are only inspectors, and inspectors bill by the hour.”

The privileged-employee rule

- **Plumbing first**: records, orders, documents; a stale page goes out at scale
- Its job **is** the lethal trifecta: safety comes from the scoping
- **Closed scope**, with the exits designed before go-live
- Start **internal and low-stakes**; carry the plumbing outward

Three jobs in one rule

A **defined role** the system makes impossible to exceed

Scoped access, per task and per duration

An **audit trail**: which prompt, model and configuration made each decision

Chapter 23, adopting a survey's phrase: "treat the agent like a privileged employee" · "Watch outcomes, not uptime."

Predict before the next slide · one minute in pairs

The clerk moves real money in real systems. The analyst only writes documents.

The finance lead asks: for which of the two can you tell me, before the pilot, how bad its worst error can get?

Chapter 23, “Framing Return on Investment Against Risk” · the chapter’s budget meeting

Two axes, and the line item the spreadsheet forgets

Return against risk, checking first

- Value of the work **minus the cost of checking it**, then the map
- Analyst: worth the machinery **and** reviewed before anything irreversible
- Refuse: a dangerous wrong fact that **nobody checks**
- Clerk: volume, a **baseline** recorded before deploying
- Deeds can be **capped**; words cannot

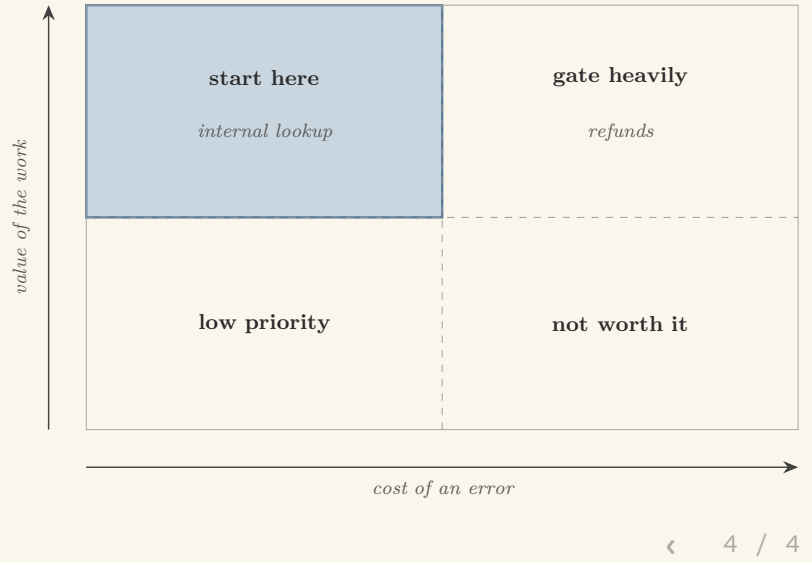


Figure from Chapter 23

Chapter 23 · “Deeds fail loudly at a size you chose. Words fail silently at a size the future chooses.”

Part 2 · Chapter 24 (full book)

The control room

Predict before the next slide · one minute in pairs

Marco sees the agent's confident summary with a green check beside it and approves in the time it takes to scroll. A mistake he could have caught in thirty seconds ships under his name.

Lena re-derives every figure by hand, finds nothing wrong for a week, and switches the agent off.

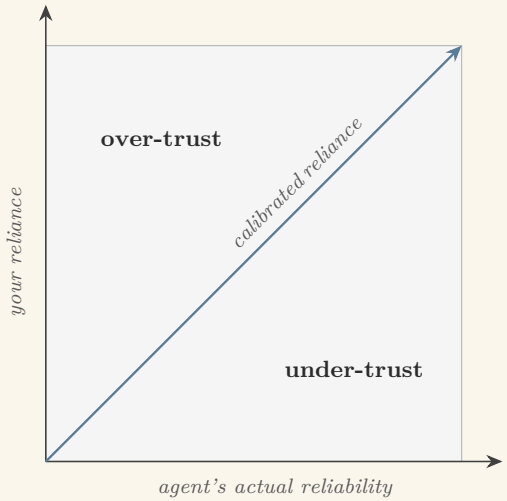
Same agent, same tasks. Who is misusing it?

Chapter 24, opening · the chapter's two colleagues, named for the slide

Not more trust: trust that tracks the record, task by task

Appropriate reliance

- **Over-trust** → misuse: its failures ship
- **Under-trust** → disuse: the saving evaporates
- Judge an interface in **both directions**
- A confidence percentage is **self-report** in a costume
- A cue **at the moment** of mismatch beats an always-on gauge



‹ 4 / 4 ›

Figure from Chapter 24

Chapter 24 · “The goal is calibration, and calibration cuts both ways” · “Information always on screen becomes wallpaper.”

Chat is the right surface for handing work over; supervision needs random access

A board, not a scroll

Station	The question it answers
Run list	which agents need me?
Approval queue	what waits for a signature?
Trace	what happened?
Evidence panel	what proves it?
Recovery controls	how do I undo?

- **Show your work**: what it read, did, changed, why it changed course, what proves it
- Not the reasoning stream: **curate for audit**
- Progress as **phase and reason**; waiting and blocked surface first
- “Done” arrives **holding its proof**

Chapter 24, quoting a published essay on supervision surfaces: “The operator needs a board, not a scroll.” · Chapter 24: “Design for the glance.”

Minute seven of a fifteen-minute run, and you learn it is building against the wrong API

A wheel, not an ejector seat

`checkpoint and inject` + `an editable plan` + `a steering inbox`

- A note lands at the **next safe boundary**; correct work survives
- Failure is a **visible state** with a repair path on it
- One false “done” resets trust to **under-trust**

Rationed alarms

Read-only: never surfaces. Medium: a **queue of typed review objects**, batched. Top tier only: an interruption.

Or approve **the plan** once.

Chapter 24, quoting a practitioner’s essay: “The abort button is the wrong affordance.” · “Steering is a transport-and-state problem wearing a UX costume”

The human’s contribution keeps moving up: from doing the work to directing and checking it

Directing, and staying in

Delegable

Verifying

Reading evidence, signing for consequences. A surface can make it cheap; it must not make it **vestigial**.

Only done or lost

Participating

The understanding that steers: the next idea, the better decomposition, the sense that an approach smells wrong.

A surface that only displays is half a design: add the walkthrough beside the diff, the questions before the signature

Chapter 24, quoting the supervision-surfaces essay of slide 15: “A supervision surface becomes worthy when it reduces work without reducing responsibility.”

Part 3 · Chapter 25 (full book)

The pattern book

A litigator, a biologist and a market analyst run the same loop

Point-task or recipe?

Point-task

A single, bounded request with a known output: one reply, one extraction, one summary

Useful, and nothing to reuse

Transversal recipe

A reusable shape of work that **recurs across industries**, **composes** into an ongoing capability, and **maps onto patterns** you already know

The test: swap the industry. Do only the nouns change?

Chapter 25 · “swap the industry, and if only the nouns change, the recipe is transversal.”

Week 6’s autonomy dial, seen from the product side

The autonomy slider

Tight end

A human **in** the loop: inside the run, approving as it goes

Loose end

A human **on** the loop: above it, reviewing what a run produced

- Set per use case; it **starts tight** and earns its way loose, on evidence
- Build more **augmentation** than autonomy: a fast generate-and-verify loop
- Fast verification is what **buys** the loose settings
- The demo–product gap is **pass@k against pass^{k}** (week 8)

Chapter 25, quoting a widely watched talk: “Demo is works.any(), product is works.all().” · “It starts tight, and it earns its way loose”

Eight plans, three bands, one load-bearing element in every one: the verification signal

Eight recipes on the slider

- **Deeds, behind gates:** digital coworker, unstructured-to-structured, queue triage, ambient watcher, self-maintaining knowledge base
- **Drafts, a human decides:** deep-research analyst, premortem and red team
- **Sandbox only:** autoresearch

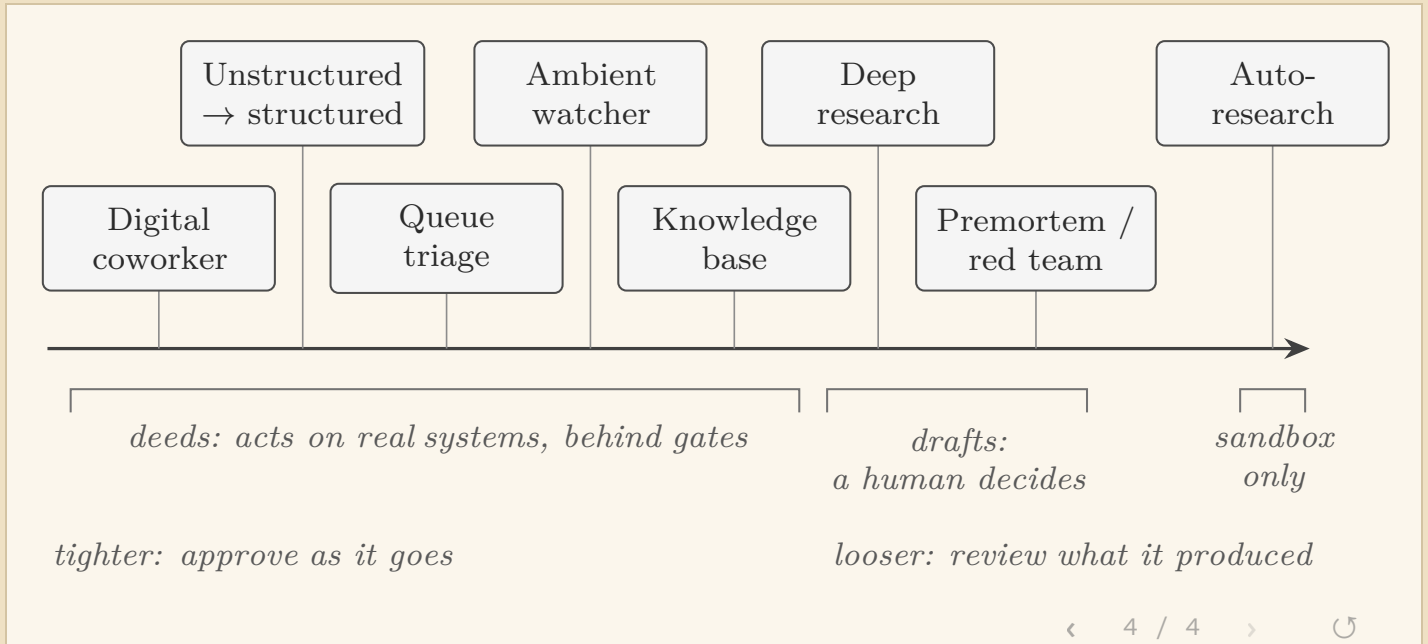


Figure from Chapter 25

Chapter 25 · “What the catalog adds is names, and names are operational”

Where a recipe sits is decided by what it touches and how fast you can check it

Three placements that teach

Far end

Autoresearch

One editable file, a scripted score, a fixed time box, all of it **disposable**

Per action

Coworker

The password reset runs free; the account closure **waits for a signature**

Advisory

Deep research

The label is the safety, and holds only if **someone inspects**

Recipes compose: the watcher feeds the triage, the triage feeds the coworker. The gate goes where the deed is.

Chapter 25 · “full autonomy is a fact about the sandbox, whatever the model’s competence.” · “the gate belongs at the consequential act, wherever in the chain it lands”

Part 4 · Chapter 27 (full book)

The map and the weather

Open your news feed and count the announcements

The map and the weather

Weather · changes weekly

Launches and shutdowns, reshuffled leaderboards, this quarter’s essential tool and next quarter’s cautionary tale

Map · changes like a coastline

The questions every agent system must answer, the layers those answers fall into, the principles that settle the trade-offs

This course taught the map, which is why it never taught a vendor’s syntax. A map cannot tell you whether to pack an umbrella.

Chapter 27 · “nothing in this chapter forecasts the weather.”

Predict before the next slide · one minute, on paper

Two services guard one seat with a lock. Each reads the lock, sees it free, and writes its claim. Every unit test passes.

Can both end up holding the lock? Would your tests have caught it?

Chapter 27, “Pairing Generativity with Deterministic Verifiers” · the chapter’s design problem

The top of the verifier ladder, paired with a tireless drafter

Generator and gate

- A test runs **one history**; a model checker walks **every reachable state**
- An **invariant** must hold in all of them; a violation comes back as a **counterexample trace**
- The model drafts the spec, invariants, property tests, proof
- A **deterministic checker** decides; the failure returns to the model
- Hallucinating gets cheap: the gate **cannot be charmed**

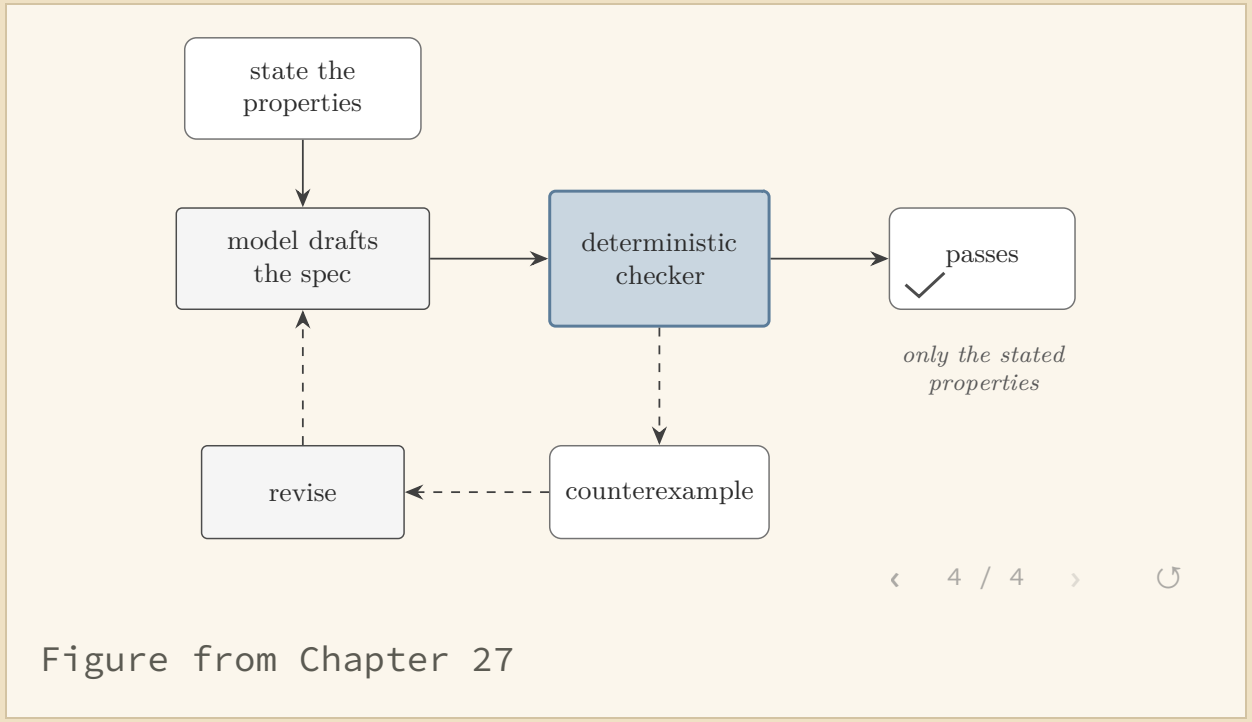


Figure from Chapter 27

Chapter 27 · “Testing samples the possible histories of a system. The bug lives in a history you never sampled.” · “Let the world find; let the model fix”

You do not need a proof assistant to start

Property-based tests as a gate

`types, schemas` → `property-based tests` → `model checking` → `proofs`

- State a property for **all inputs**; the tool generates cases and **shrinks** a failure
- The model **proposes** properties; you prune the silly ones
- The framework, not the model, **decides**

Four limits

State spaces explode, and the model will not keep them small

A passing spec can be the wrong spec

It may recite the textbook system, not yours

Experts still get far more out of it

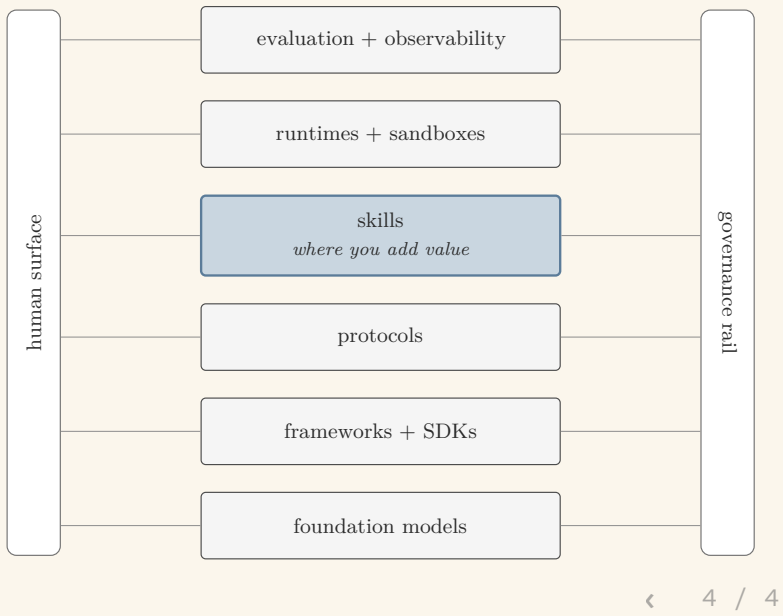
Chapter 27 · “let the framework, never the model, decide which survive.” · “a spec that passes can be the wrong spec”

Strip the brand names off any agent stack

Six durable layers, two rails

1. **Models**: what computes the next step?
2. **Frameworks**: who runs the loop?
3. **Protocols**: how do parts connect?
4. **Skills**: where does packaged know-how live?
5. **Runtimes, sandboxes**: where do actions run?
6. **Evals, observability**: better or worse?

Two rails beside every layer: the **human surface** (where a person steers and approves) and **governance** (who may do what, checked where)



‹ 4 / 4 ›

Figure from Chapter 27

Chapter 27, quoting an essay on “AI operating system” claims: “If permissioning lives only inside prompt instructions, you do not have access control—you have a polite request.”

The portable core

- **Yours**: prompts, skills, tool contracts, eval sets and the traces they grew from, memory files, goal functions, guardrail policies
- **Rented**: the model, the framework, the runtime
- Three lock-ins: model, orchestration, data; insure each with a **thin seam**
- Scale the insurance to how long the system must live

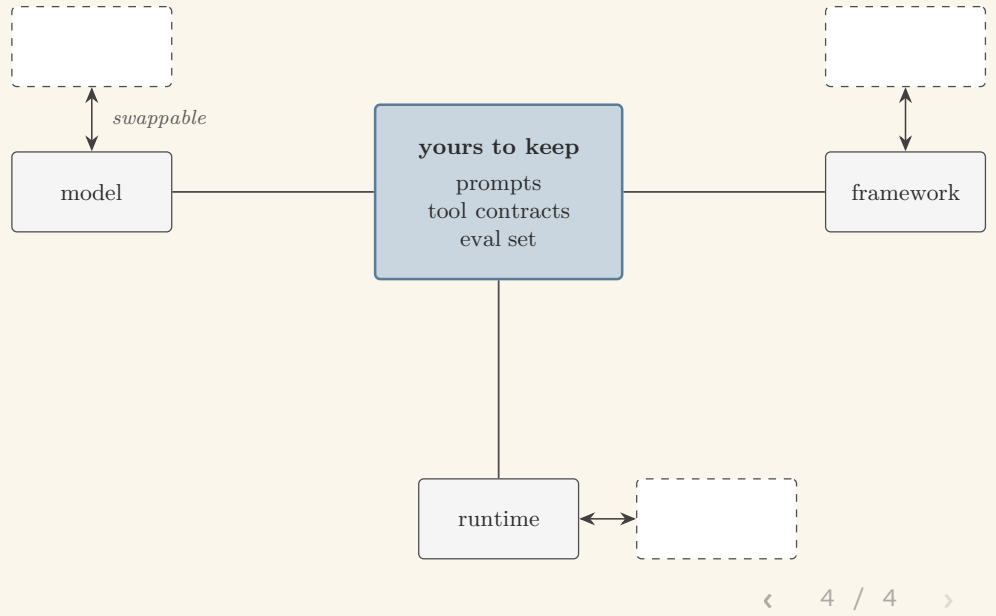


Figure from Chapter 27

Chapter 27 · “never store what you own inside what you rent.” · “Rent the engine. Own the judgment. Keep the receipts in your own drawer.”

No predictions: directions you can plan around without betting on a date

Directions, and four habits

- Larger units handed over; the human **higher up**
- More capability for the cost; keep the seam thin
- Longer task horizons: the **harness** matters more
- Capability arrives **packaged**
- What does not move: **verification** stays scarce

Keep the map current

Read **primary sources**; the caveats are the payload

Build small things against new products

Keep a **private benchmark** of tasks models fail

Principles over products

Chapter 27 · “Directions are map features. Arrival times are weather.” · “A launch post is a claim. A task from your menagerie, falling, is a signal.”

In class · teams of two or three · 24 minutes · three browser tools

Price it, score it, place it

1. **A · two hires (8 min).** The analyst and the clerk on the ROI versus risk map
2. **B · a scorecard that flatters (8 min).** Re-mark another team's verifiability scorecard from its evidence
3. **C · your capstone (6 min).** The map, the recipe picker and your own scorecard

The prefilled links and the evidence are on the lecture page. Debrief, 2 minutes

Tools after Chapters 1, 23 and 25 · each tool's readings are its own, labeled on the tool · the two hires and the other team are invented for the exercise

The capstone · the brief in the syllabus

Demo day

Twelve minutes a team, or ten in a large class

Three screens, in this order

1 · open on

The eval dashboard

Per-task rates over at least three runs, a pooled rate with its interval, pass@k **beside pass^{k}**, capability apart from regression

2 · then

A trace

One the **panel picks** from your sorted run list; walk it to the first wrong span, or say why none went wrong

3 · end on

Your scorecard

Every “yes” **points at an artifact** the panel can open

The capstone brief · agent verifiability scorecard · Chapters 15, 16 and 25

The rules are grading conditions, not etiquette

What counts on demo day

- **No cherry-picked runs.** You do not choose the run you show
- **Results are distributions.** Grading reads pass^k, never one lucky run
- A live run is an illustration, read **against your reported rate**; on its own it counts toward neither the eval set nor the trace review
- Every member **explains one part**; one, named by the panel, **changes one thing live**. The others change theirs in a short follow-up

The capstone brief · the course's capstone rubric · "Demo is works.any(), product is works.all()." (Chapter 25)

A look back, without the syllabus

Thirteen weeks, one question

	Week 1	Week 13
What you hand over	one model call in a loop	a recipe running on a stream, overnight, at scale
Where you sit	inside every step	on the loop, at a board, signing
What you pay for	tokens	tokens, and the checking
What you ask	what signal tells you it worked?	

Chapter 1 asked it first; Chapter 27 ends on it

Before demo day, and after the course

What is left to do

For the demo

- Part C's three exports, finished
- The run list **sorted**, traces ready to open
- A premortem of your own demo, and its mirror
- The reading quiz on this week's chapters

To keep

- Start a **private benchmark**: five tasks today's models fail
- App. C, Further Reading `full book`
- The glossary `free`

This week's reading: Ch. 23, Ch. 24, Ch. 25 and Ch. 27 (full book) · no paid API needed

Five things to keep

- No compiler for facts: **partial substitutes**, checking matched to stakes
- Subtract the **checking** first; deeds can be capped, words cannot
- Recipes start **tight**; the gate goes where the deed is
- Design for **appropriate reliance**: a board, evidence at the claim
- Rent the parts, own the judgment, let a **gate** check the generator

Engineering AI Agents · Week 13

What signal tells you it worked?

Slides from the teaching kit of *AI Agents, Engineered* by Enrique Gutiérrez, CC BY 4.0 · aiagentsengineered.com/teach/

Reuse, adapt and translate with attribution · creativecommons.org/licenses/by/4.0/

