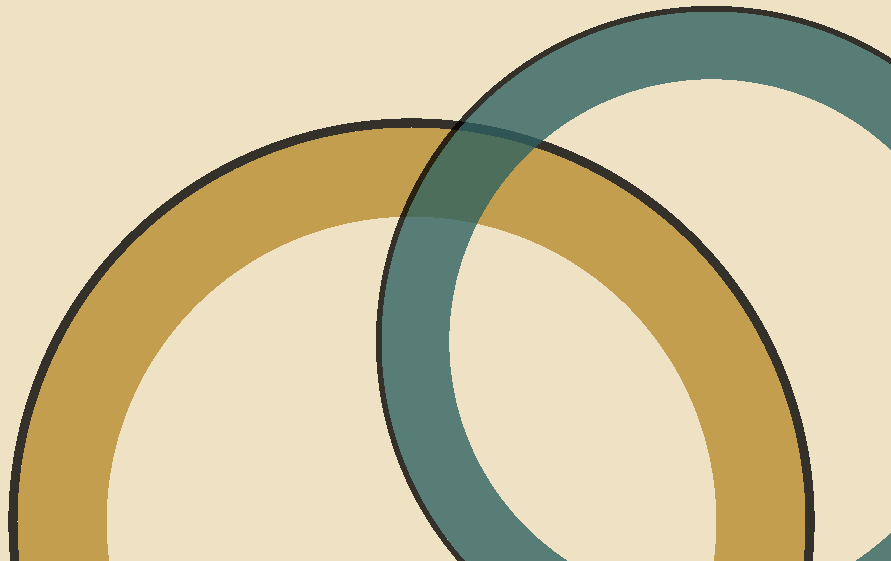
Engineering AI Agents · Week 6

Multi-agent, oversight, the outer loop, and when not to

Splitting the work · signing for it · running it overnight · deciding whether to buy it



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

The task: “Summarize the competitive landscape for managed vector databases.”

Write down the subtasks you would hand to a team of workers.

By the end of today you can

Four objectives

1. Decide **single or multi-agent** with one question; write a **brief** with four fields
2. Sort tools into **four consequence tiers** and write the **gate policy**
3. Specify a **goal function** with a hard cap and a **kill switch**
4. Price a request on the **ladder** with the **four-line quote**

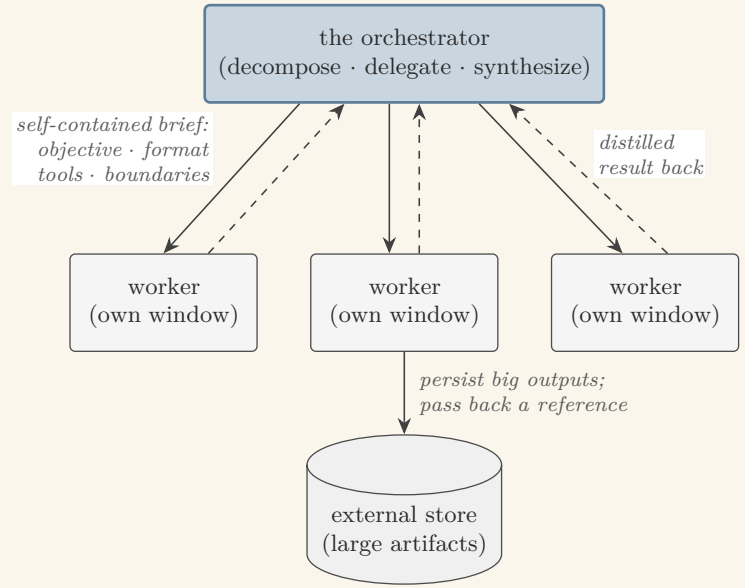
Part 1 · Chapter 11 (full book)

Multi-agent systems

The dividing itself takes judgment, exercised at runtime, on this input

The orchestrator–worker pattern

- A lead **decomposes**, workers run in their own windows, the lead **synthesizes**
- Week 5's fan-out: *you* wrote the list at design time
- Buys speed and clean desks
- Costs tokens: ~4× a chat for one agent, ~15× for multi-agent



Large outputs go to an external store, so only a reference passes back through the lead.

‹ 4 / 4 ›

Figure from Chapter 11 · ~4× and ~15×: one team's measurement of its own system, illustrative

Predict before the next slide

An orchestrator hands three workers the same brief: “research the semiconductor shortage.”

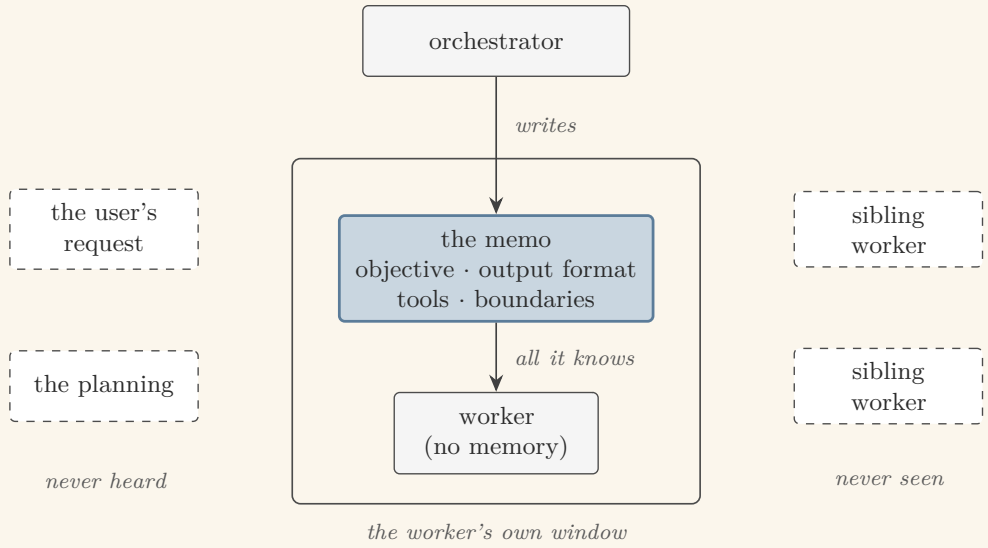
What comes back?

Chapter 11, “Focused, Self-Contained Worker Context” · from one production team’s published report

Delegation quality is the variable that most separates systems that work from those that embarrass you

The brief: four fields

- **Objective**: the one thing this worker produces
- **Output format**: the shape it comes back in
- **Tools and sources**: which tools, where to look
- **Boundaries**: what belongs to a sibling
- The test: a ticket for a contractor who has never seen your project and **cannot ask questions**



The worker is exactly as good as the memo that summoned it.

‹ 4 / 4 ›

Figure from Chapter 11

With five agents you own a distributed system,
and its bugs live in the seams

Four seams, one rule

Seam	What it looks like
Duplicated work and gaps	One vague brief, several private readings, an uncovered flank
Conflicting implicit assumptions	Every piece is right; the pieces do not compose
The game of telephone	A summary of a summary; the load-bearing caveat shaved off
The laundered return	An injected instruction, delivered in a colleague's voice

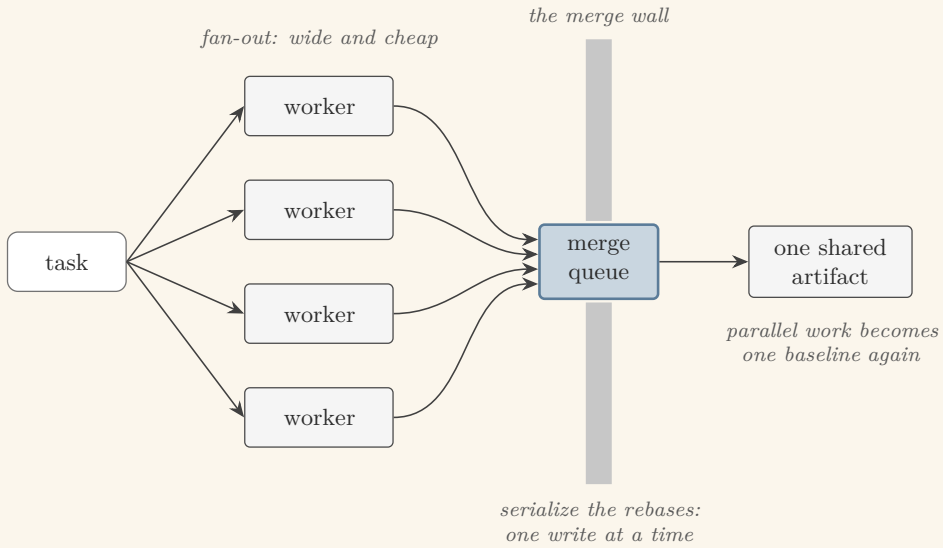
Keep writes single-threaded: one agent owns the artifact; everyone else advises.

Chapter 11, "Coordination Failure Modes" · 1,600+ traces, 14 modes in 3 clusters: Cemri et al. (2025)

Map is cheap and orderly; the reduce can be arbitrarily complicated

The merge wall

- Workers branch from a baseline that each of them can change
- A **merge queue**: serialize the rebases
- **Triage**: some work is inherently serial
- **Rhythm**: swarm for a while, then pause and serialize



Only single-file writes turn parallel branches back into one shared artifact.

‹ 4 / 4 ›

Figure from Chapter 11

Chapter 11 · “Adding agents adds hands and coverage; it does not add judgment.”

The verdict is a procedure

"Does this work decompose into branches that can run without seeing each other?"

Yes

Broad research, wide review, evaluation sweeps: the orchestrator is on the table

No

Shared state, silent decisions, an order that matters: keep one continuous context

If you split: single-threaded writes, briefs for a stranger, tracing from day one, the bill measured

Chapter 11, "The Merge Wall and the Single-versus-Multi Verdict"

Part 2 · Chapter 12 (full book)

Oversight and autonomy

Check · pairs · one minute

Which of five tools wait for a signature?

`look up an order` `draft a reply` `send the reply` `issue a refund`

`delete a customer record`

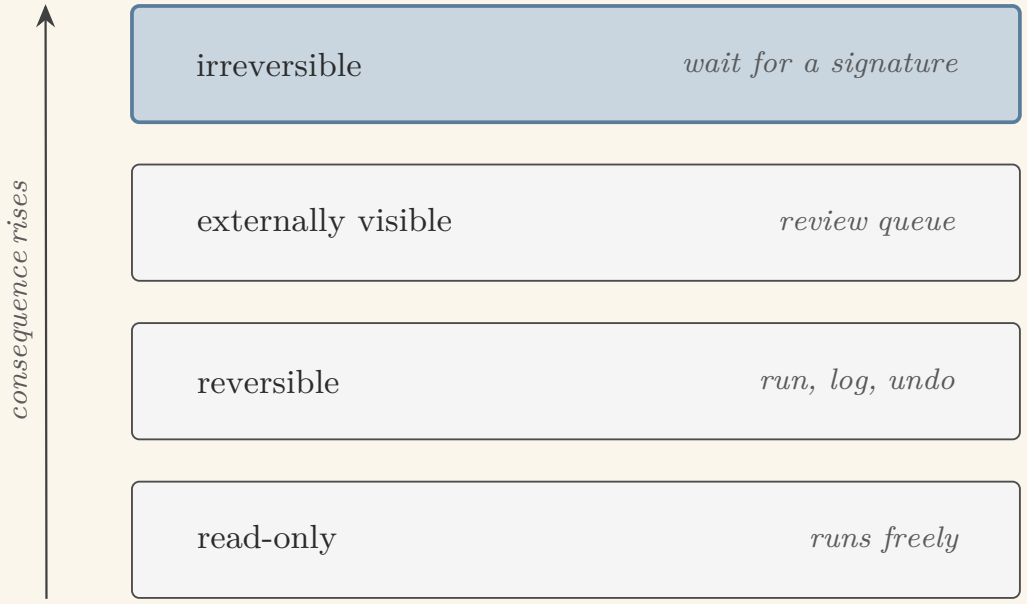
Then: what did you consult to decide?

Chapter 12, “Approval Gates and Escalation” · the chapter’s support agent

Grade the action by what it costs when it is wrong

Four consequence tiers

- **Read-only**: runs freely
- **Reversible**: runs, if logged to undo
- **Externally visible**: review queue
- **Irreversible**: a signature, every time
- The model's **confidence plays no part**



‹ 4 / 4 ›

Figure from Chapter 12 · claimed 90% confidence, ~75% real: one production analysis, illustrative

After the model has chosen, before anything runs

The gate lives in your code

`model proposes` → `validate` → `gate` → `execute`

- Fires on **what the action is**
- Escalation fires on **how the run is going**: repeated failure, out-of-policy input, signs of injection, a budget nearly spent
- May it wait? Queue it. Irreversible or suspected injection? Block.

The handoff

The action in plain language · why · what it touches, and can it be undone · before → after · approve · reject ·

reject with edits

Chapter 12 · “Send a decision, not a transcript.” · “Over-gating does more than annoy. It defeats the gate.”

Four hundred lines across nine files, tests green, twenty minutes later: approved

Review intent and outcomes, not lines

Criteria

“Done” stated before the run

Scope

Declared; anything outside is a flag

Walkthrough

What changed and why, in order

Evidence

Proof a stranger could check

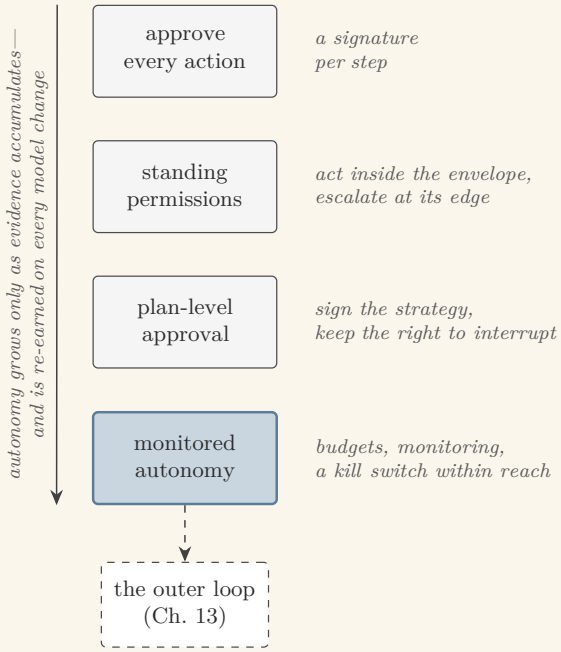
Cheapest of all: review the **plan** before any code exists. Verify what you can delegate; understand what your next decisions depend on.

Chapter 12, “Reviewing Intent and Outcomes, Not Lines” · *review theater*: diligence performed at a volume where it can no longer be real

How much an agent may do between moments of your attention, set per task

The autonomy dial

- Moves **on evidence**, in both directions
- Start every new task class tight
- Re-earn the setting after **every model upgrade**
- The same person runs both ends on the same day
- Interruption is only cheap if resuming is



A task moves down the dial only on evidence, re-earned on every model change.

‹ 6 / 6 ›

Figure from Chapter 12

Part 3 · Chapter 13 (full book)

Writing the outer loop

Predict before the next slide

The same prompt file, piped into a coding agent, forever, on a fresh project, overnight:

```
while :; do cat PROMPT.md | agent ; done
```

What is there in the morning?

Chapter 13, “A Short History of the Loop” · generalized from a practitioner’s post

“Deterministically bad in an undeterministic world”

Three lessons from a line of shell

State lives on disk

Spec, plan file, instructions file, the commit history: the model forgets, the files do not

Windows stay fresh

One task per pass; no pass lives long enough to rot

Verification is a gate

Commit only when the compiler, types, tests and linter say yes

Failure catalog: building a feature twice after wrongly concluding it was missing · stubs that pass the checks · paying to re-read the spec every pass

Chapter 13 · the technique’s nickname and slogan are its author’s; the phase loops that followed made the same three lessons explicit

Write the stopping condition for “make the app better.” Then look at what you wrote

The goal function

- A **persistent goal**, held outside the model, on disk
- A **stopping condition** a machine can evaluate, after every pass
- An **oracle** outside the work; separate the checker from the maker
- A **hard iteration cap** beneath it, and a budget

“A loop is exactly as trustworthy as its oracle.”

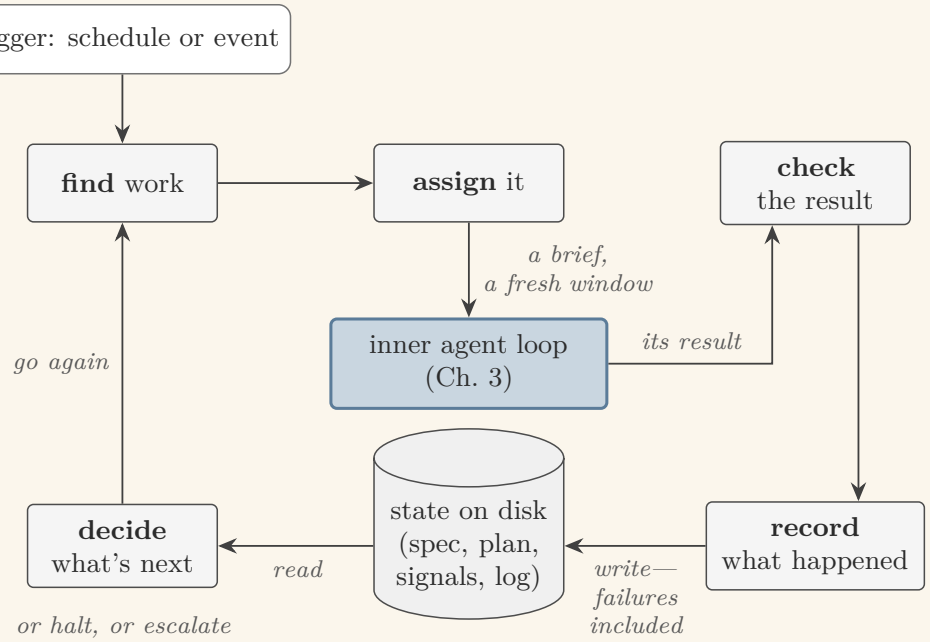
Is “done” machine-verifiable? Loop. If not: a person, or a calibrated judge, and a weaker guarantee

Chapter 13, “The Goal Function”

Strip any outer loop and the same five verbs are underneath

Find · assign · check · record · decide

- **Find**: a trigger, then candidate work
- **Assign**: a fresh window, a brief for a stranger
- **Check**: the mechanical gate, then a fresh read-only verifier
- **Record**: failures included
- **Decide**: halt, go again, or escalate



Decide reads that state and goes again, halts, or escalates to a human.

‹ 5 / 5 ›

Figure from Chapter 13 · "Build the loop. Stay the engineer."

A loop that works unattended also makes mistakes unattended, at scale, on your bill

Two debts, five brakes

Verification debt

A green gate is a fact about the checks you thought to write

Cognitive debt

Verified work that has outrun your understanding

- **Budgets**: iterations, money, time; per pass and per loop
- **Stuck detection**: the same check failing pass after pass
- **Least privilege**, sandboxed; audit the trifecta per loop
- **Human gates** at the blast doors
- A **kill switch**, tested before the first unattended run

Chapter 13, “Keeping the Loop Safe” · “A kill switch discovered broken during the fire is a design document for the next system.”

Part 4 · Chapter 14 (full book)

Choosing your approach: when not to build an agent

Commit to four answers before the next slide

The cheapest thing that would do the job well

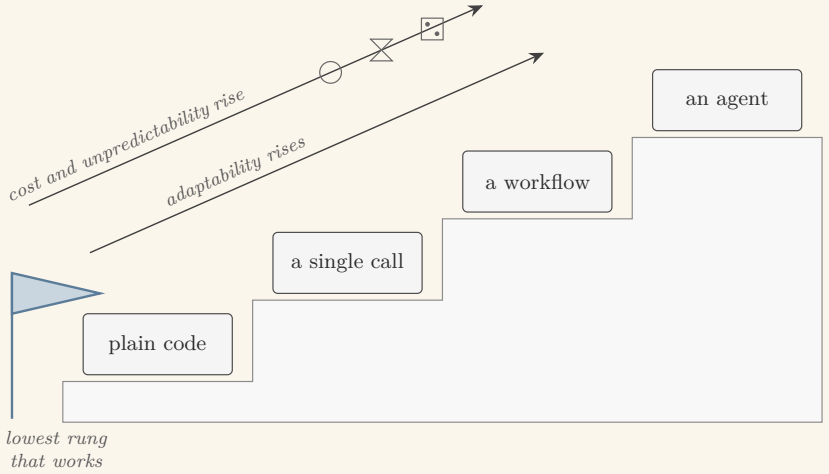
1. Flag every wire transfer above fifty thousand dollars for manual review
2. Sort incoming support tickets into five categories and route them
3. Find out why the nightly build started failing, and fix it
4. Automate our customer support

Chapter 14, opening · the chapter’s four requests

A rung's true quote has four lines; the fourth is the one engineers leave off

The ladder, priced

Line	What grows up the ladder
Build	Spec → eval set → decomposition and gates → a harness
Run	Free → per call → costs add → costs multiply
Wrong	Same bug every time → words → contained → deeds
Know	A unit test → sampling forever → per-step tests → a second construction



‹ 4 / 4 ›

Figure from Chapter 14

Chapter 14, "The Ladder Revisited" · "The sticker price of an agent is a prompt."

The litmus: can you draw the flowchart before the request arrives?

Signs, both ways, and a trap

Down the ladder

- The steps can be enumerated
- Correctness must be guaranteed or audited
- A person waiting, or high volume at low margin
- Lookup, classification, transformation
- High stakes, no cheap verification

Up the ladder

- Open-ended; judgment at branches you cannot list
- **And** a machine-checkable success criterion, feedback at each step, a place for a person

Agent-washing: the label sells. Ask: does the model decide the next step, or does code?

Chapter 14, “Signs You Need One, Signs You Don’t” · the ladder is climbed in both directions

In class · teams of three · 22 minutes · two browser tools

A gate policy, then a brief

1. **A · the gate policy (10 min).** Open the consequence tier classifier (seven helpdesk tools loaded, questions blank). Answer each tool from its description on the lecture page. Try the tightest dial: which row changes? Back to standing permissions; copy as Markdown
2. **B · the brief (8 min).** Open the multi-agent failure explorer with the one-line brief loaded; record its reading. Write the brief for the worker who lists product changes; check it; copy as Markdown

Debrief, 4 minutes: one warning and its fix; one brief read aloud as a contractor would read it

Tools after Chapters 11 and 12 · the question order and the brief's word patterns are the tools' own · scenarios invented for the exercise

Lab 5, the proposal, and reading

Lab 5 · a gate and a loop

- A **gate in code** for the two top tiers, with the handoff
- An **outer loop** over a checklist on disk: five verbs
- A machine-checkable stop, a **cap**, a tested **kill switch**

Project proposal (teams)

The problem, its rung with the four-line quote, the verification signal

Reading for week 7

- Ch. 15, Observability and Debugging
  `full book`

This week’s reading: Ch. 11 · Ch. 12 · Ch. 13 · Ch. 14 (full book) · glossary: App. B (free) · no paid API needed

Five things to keep

- Split only work whose branches **can run without seeing each other**; writes stay single-threaded
- A worker is exactly as good as its **brief**: objective, format, sources, boundaries
- Gate by **consequence**, in **code**; send a decision, not a transcript
- An outer loop is as trustworthy as its **oracle**; cap it, and test the kill switch
- Price the rung with **four lines**; the fourth, knowing, is the one left off

Engineering AI Agents · Week 6

Build the loop. Stay the engineer.

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/

