

A HUMAN TEXTBOOK FOR INSPECTABLE SOFTWARE WORK

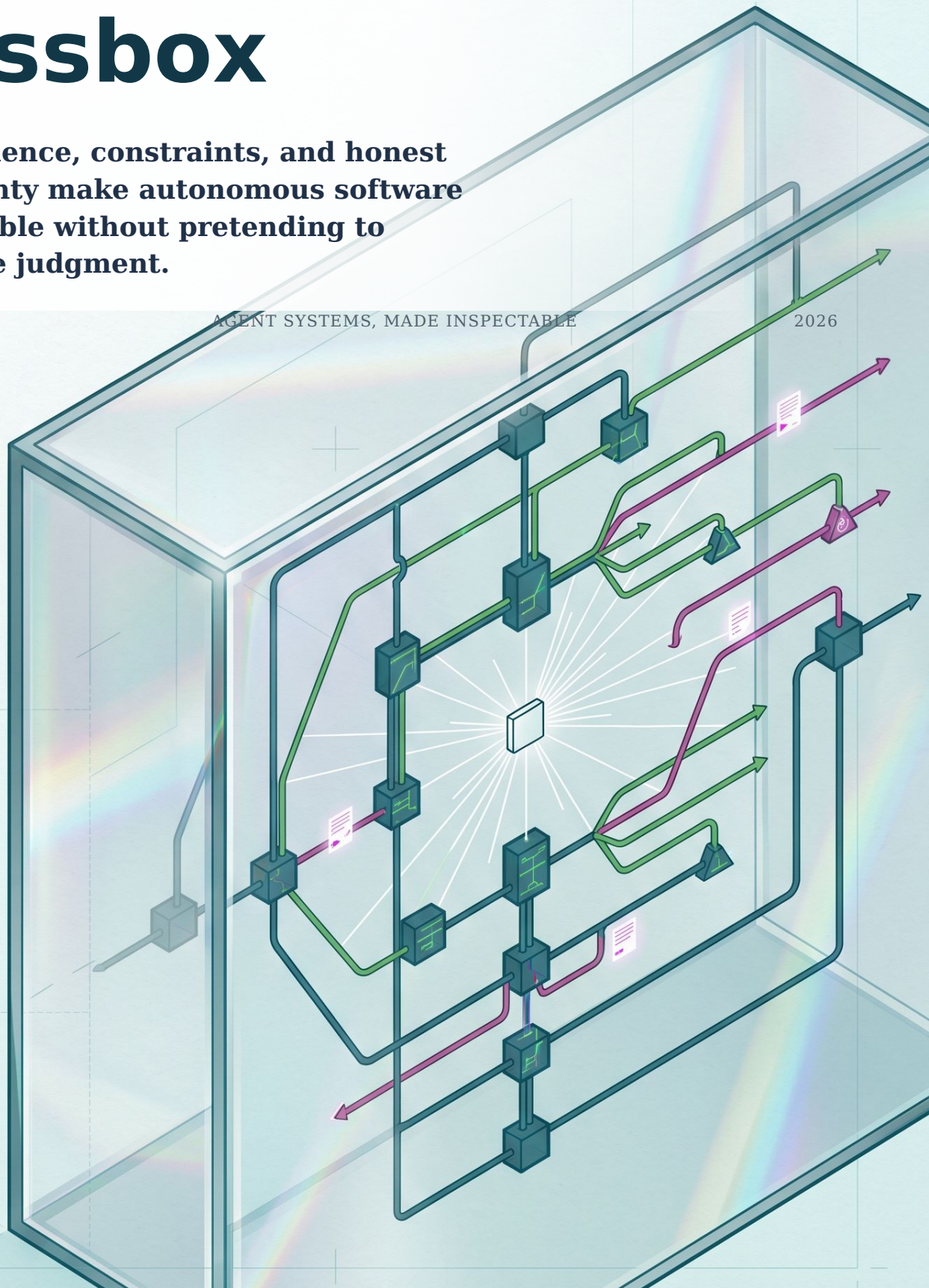
Inside Huru's Glassbox

How evidence, constraints, and honest uncertainty make autonomous software work visible without pretending to automate judgment.

HURU / LETTUCE

AGENT SYSTEMS, MADE INSPECTABLE

2026



 LETTUCE / GLASSBOX SERIES

Inside Huru's Glassbox



Agents are not outside the systems they build. Their work, evidence, uncertainty, and constraints belong inside the glassbox too.

The same must be true of the humans who direct, interrupt, approve, and remain accountable for them.

 LETTUCE / GLASSBOX SERIES

Inside Huru's Glassbox

Evidence, constraints, and human judgment
in agent-operated software systems

HURU / LETTUCE · 2026

PREVIEW MANUSCRIPT

This book is itself a glassbox exercise. Humans set its purpose, challenge its claims, and remain responsible for what it teaches. Agents research, draft, verify, revise, and preserve the work through Lettuce.

The manuscript, figure sources, evidence notes, editorial decisions, and reproducible build procedure live with the product documentation. Generated imagery is identified by source metadata; product claims are labelled by status and bounded by the evidence inspected.

No ISBN has been assigned. This preview is not a statement that every described capability ships or that every open research question has been resolved.

Contents

• How to Read This Book.....	9	• Parallelism, Quorum, and Bounded Convergence.....	85
• Preface: Why Open the Factory?	13	• Composition Without Opacity	89
• The Running Case: <u>CHECKOUT RECOVERY</u>	15	• The Graph Pattern Catalog	95
• The Factory Problem.....	19	• Probability and the Harness	105
• The Glassbox Contract.....	23	• Human Authority and Decisions That Survive.....	111
• The Boundary of Knowing	27	• Failure Modes That Survive Green....	117
• A Map That Can Say Not Yet	33	• Delegated Semantic Authority.....	123
• Evidence That Can Age.....	39	• An Honest Research Horizon.....	131
• The Loop That Remembers.....	43	• Reading the Result Months Later.....	135
• Method, Evidence, and Enforcement ..	51	• Future Directions: From Glassbox to Supervised Factory	137
• Graphs Are Contracts Before They Are Pictures.....	59	• Appendix: A Working Glassbox Vocabulary	147
• The Shape Catalog.....	65	• Appendix: Verified Product Receipts .	151
• Run-Cases, Carriers, and the Ledger of an Attempt	79	• Appendix: Claim and Status Ledger ..	155

How to Read This Book

This is a textbook and a case study. It uses Lettuce to make general arguments about agentic software work, then tests those arguments against a real product surface. The visual vocabulary below makes the kind of statement visible before the reader evaluates it.

OPERATING PRINCIPLE

A general rule that the reader can apply beyond Lettuce. It is an argument, not proof that the current binary implements every implication.

CONTRACT

A precise boundary: what a mechanism is allowed to claim, what it refuses, and what remains outside its authority.

LETTUCE IN PRACTICE

A source-backed product example. It must name whether it is a shipping behavior, doctrine/design, or research hypothesis.

FAILURE MODE

A way a metric, workflow, or agent can create a false appearance of progress. These boxes are deliberately more prominent than ordinary notes.

REFLECTION

A question for the reader or team. It does not have a pre-authorized answer and should not be converted into an automated gate without a defined oracle.

Every callout has both a written label and a line icon. The icon is a navigation cue, not a second claim: a measured axis means a principle or reader keep, a checked frame a contract, a case a concrete practice, a triangle a failure mode, a clock face a reflection, and an open prompt a question that still needs an owner.

One store, two consumers

Lettuce has one durable project state but two deliberately different ways of consuming it:

Consumer	Primary surface	What it does	What it must not assume
Agent	CLI or HTTP, machine envelopes, skill, usage, wiki	Orients from the map, claims work, follows contracts, records runs and evidence, files discoveries, and advances or receives refusals.	That a successful mutation proves semantic quality or grants authority.
Human	Web board, graph and run visualization, timelines, replay, evidence views, escalation and decision packets	Observes the declared frontier and history, challenges claims and bars, investigates exceptions, and decides where accountable judgment is required.	That a polished board or completed graph proves the project is good.

These are not two stores and should not become two stories. The board an agent uses to orient and the board a human sees must project the same underlying map. The run an agent advances and the route a human replays must fold from the same events. The evidence an agent links and the artifact a human inspects must keep the same provenance.

⚡ AUDIENCE CONTRACT

Agents operate Lettuce. People observe, challenge, govern, and decide through its shared visual and historical surfaces. A human may use diagnostic commands in an exceptional investigation, but command operation is not the book's reader workflow.

Product evidence and figures

The book may name a command when that name grounds a shipping claim, and its source fixtures may contain complete invocations used for verification. Those commands are evidence exhibits, not a workflow for the human reader. Agent operation belongs in the embedded skill, generated usage, and wiki. This book explains the resulting behavior, refusal, record, and visible consequence.

Every figure has a title, a caption stating the claim it illustrates, and a source status. A diagram may clarify a model; it does not turn the model into a shipping feature. Figures about the current product must cite a reproducible source. Figures about doctrine or research must say so in their caption.

Quotations name their source and are kept short. The surrounding text distinguishes what the source says from the book’s interpretation. Case studies preserve failure, uncertainty, and reversal when those are part of the lesson; they are not success-story advertisements.

Contract cards: how to read a shape

Whenever the book introduces a graph, station, gate, carrier, or other coordination object, it will use a **contract card** before relying on the object in prose. The card is a fixed visual vocabulary, not a compact marketing summary. Read its fields in this order:

S Reading key: a coordination object	
S SHAPE The structural role or geometry. It says what kind of station the graph contains.	C CONTRACT The narrow fact, handoff, or state boundary the object is allowed to establish.
B BEHAVIOR What the station or mechanism does when the declared contract is met.	X CONSTRAINTS What is refused, bounded, versioned, or otherwise prevented from becoming implicit.
A AGENT INSTRUCTION The procedure expected of an agent. It guides work; it is not proof the work happened well.	H HUMAN BOUNDARY The judgment, authority, or decision that remains outside the object's mechanical reach.

The letter marks are consistent throughout the book: **S** is structural shape, **C** is a narrow contract, **B** is behavior, **X** is a constraint, **A** is agent instruction, and **H** is the human boundary. A card that lacks a human boundary is incomplete whenever the object can influence a consequential claim. Conversely, a human-shaped station is not enough: the card must still say what the person is authorized to decide and what record survives that decision.

For a Lettuce graph, the **Shape** field must be backed by `lettuce graph-def viz`, not a hand-drawn approximation. The book may render that deterministic Mermaid output as a static SVG for PDF readability, but it keeps the graph JSON fixture and generated Mermaid source beside the image. The SVG explains the same topology; the generated source remains the authority for shape, labels, and edge order.

Preface: Why Open the Factory?

Software delivery has always mixed mechanism with judgment. Compilers, tests, type systems, deployment gates, and incident checks answer narrow questions with extraordinary speed. Product fit, architectural coherence, understandable interfaces, acceptable risk, and the meaning of an exception remain slower and more contextual. Language-model agents do not remove that distinction. They make it easier to produce work at the exact moment when we most need to inspect what the work means.

This book began with an uncomfortable observation: an automated software factory can become more opaque as it becomes more productive. A task turns green. A test suite passes. Several agents agree. A graph reaches its terminal. Every signal may be accurate and the larger conclusion may still be wrong. Throughput increases the number of claims a team must interpret; it does not automatically improve the interpretation.

Lettuce is the case dissected here because it attempts a different bargain. It does not promise to manufacture correct software unattended. It gives work an inspectable address, gives selected rules a place to refuse progress, preserves the evidence and route of an attempt, and keeps unresolved judgment visible. The result is a glassbox: not perfect knowledge, but a system in which a later reader can reconstruct what was claimed, what happened, which rule admitted the next step, and what remained undecided.

A LIMIT ON THIS BOOK

Lettuce is still developing. Every current-product claim is bounded to the public surface verified for this manuscript. Design directions and research hypotheses are named as such. The book is partly an account of what exists and partly a disciplined record of what the case reveals is still missing.

The intended reader is a human observing, reviewing, challenging, or governing a system in which agents do consequential work. The agents operate Lettuce: they read its map, claim work, follow or test its contracts, and write back the receipts that orient the next cycle. The human does not need to become a Lettuce command-line operator. The human needs to read what the agents made visible in the board, graph, run history, evidence, refusal, and decision record.

You do not need Lettuce to use the central method. Ask four questions of any automated workflow:

1. What claim is being advanced?
2. What evidence supports exactly that claim?
3. Which rule can refuse the next step?
4. Which judgment remains outside the rule?

The chapters apply those questions to maps, evidence, autonomous loops, graphs, probabilistic workers, and decisions. CHECKOUT RECOVERY keeps one unresolved customer-safety question alive across every layer. If the machinery makes that question disappear without answering it, the machinery has failed the lesson. Agents operate the map; people observe and govern its visible record.

The Running Case: CHECKOUT RECOVERY

This book returns to one small fictional project. It is deliberately ordinary: an online checkout must tell a customer what happened after a payment request times out. The point is not to make checkout software the book's subject. The point is to give scopes, evidence, graphs, and decisions a consequence a reader can keep in view.

The initial claim

The product team wants to say: **after a payment timeout, a customer can tell whether to wait, retry, or check their order history without risking a duplicate purchase.**

The existing automated suite passes. It proves useful things about known request and response paths. It does not establish that the timeout state is clear on a narrow screen, that the order history reaches the same conclusion, or that the team's operational support staff can explain a pending charge. The tempting false conclusion is already available: green tests mean the checkout recovery is done.

The first map

The project begins with a modest declared boundary:

Scope	Units	Questions to apply
Checkout recovery	payment submission, status return, order history	timeout recovery, responsive experience, state clarity, evidence freshness

This is not a claim that the table is the whole product. It is a promise to examine these units under these questions before claiming this release path is ready. Later chapters will add a missing state, revise a receipt, and challenge the map itself. That is intentional: an honest map must be allowed to become less certain before it becomes more useful.

The human decision that the map cannot make

The first review finds that the timeout screen says only "something went wrong." The test receipt is real, and the initial screenshots are real, but neither answers the release question: can a customer safely retry without risking a duplicate purchase or an unexplained pending charge?

Three options are now plausible:

1. release the change because the known request paths pass;
2. hold the change until the customer-facing payment state is made explicit; or
3. release behind a narrow operational constraint while instrumenting the unresolved state and giving support a documented response.

No general test, carrier, or coverage ratio can select among those options. The release owner must weigh customer harm, rollout scope, operational readiness, and the cost of delay. That is a semantic and consequential decision, so the book treats the human as an authority with a question to answer, not as a decorative approval node.

For the teaching case, the owner chooses the second option: hold this recovery path until the timeout state and order-history explanation agree. The reason is not “the board was red.” It is that the available evidence does not support the customer-safety claim the release would imply. Later chapters will show how new evidence can challenge and supersede this choice rather than making it a silent, permanent fact.

Current product boundary. Lettuce can record task narrative, workflow requests and resolution, evidence links, and graph-run provenance. A general structured decision receipt with explicit options, authority, conditions, and supersession remains proposed product work. The case uses that distinction on purpose: recorded provenance is not yet a complete decision ledger.

The thread through the book

The story will move through five moments:

1. A passing suite and an initial screenshot set support a narrow claim but leave the timeout meaning unresolved.
2. The team declares cells, evidence bars, and a Definition of Done instead of treating the task list as the denominator.
3. An agent works a bounded task, a required receipt exposes a gap, and the map changes rather than preserving a false green.
4. A graph coordinates focused review work and leaves a durable carrier/effect trail; it does not decide whether the release is wise.
5. A human makes a release decision, later evidence changes the premise, and the project reconstructs and supersedes the earlier decision.

The case is a **teaching composition**, not a transcript of one current Lettuce store. When a chapter names a shipping behavior, it will identify the behavior as such and verify it separately. The case is here to make the reasoning concrete, not to smuggle imagined product behavior into the reader’s trust.

Keep this question active. At every stage, ask what the project can honestly say about checkout recovery now, what receipt supports that statement, and who must decide the remaining risk.

The Factory Problem

Claim status: doctrine and general engineering reasoning. This chapter does not claim that a particular Lettuce command can judge architecture, product fit, or maintainability.

? The question

Why is a passing suite not enough to let the CHECKOUT RECOVERY team declare its timeout path complete?

💡 The thesis

Automation is most valuable when it is attached to a claim with an adequate oracle. It becomes dangerous when a real but narrow signal is upgraded into a judgment about a product, a design, or a release decision.

Autonomous software work begins with an appealing simplification: give an agent a goal, a repository, a test suite, and enough time. If the tests pass, the work is done. If they fail, the agent tries again. The loop looks objective, cheap, and infinitely repeatable.

The simplification fails at the point where the available signal stops matching the desired outcome. Tests can answer many valuable questions quickly: did a known behavior change, does a request satisfy a contract, did a regression reappear? They cannot, by themselves, answer whether a change preserved the system's conceptual integrity, made the next change easier, solved the right customer problem, or turned an exception into a future maintenance trap.

That gap is not an argument against automation. It is an argument against pretending that every valuable property has a cheap oracle.

For CHECKOUT RECOVERY, the existing suite can establish that selected payment requests and callbacks behave as encoded. It cannot establish that a customer who sees a timeout understands whether money moved, whether retrying is safe, or where to find the eventual order. The first project decision is therefore not "how do we make the board green?" It is "what narrower statements can we support now, and what must remain open?"

⚡ OPERATING PRINCIPLE

Automate the claim for which you have an oracle. Preserve and escalate the judgment for which you do not.

The proxy trap

A proxy is a measurable stand-in for something harder to observe. Test coverage can proxy exercised code. Review count can proxy attention. A completed task can proxy a decision. A green board can proxy progress. Each can be useful. None is the thing itself.

The danger begins when a proxy becomes the only score that matters. A system asked only to make tests pass can add brittle special cases, weaken assertions, or hide complexity behind an interface. A team asked only to close tickets can slice work into deceptively small claims. An agent asked only to harden cells can attach evidence that exists without establishing the claim a reader inferred from it.

This is a familiar Goodhart problem: once a measure is optimized as a target, it tends to lose value as a measure. The remedy is not to discard measurement. It is to keep the claim narrow, expose its provenance, and reserve judgment for the part that cannot be made mechanical.

FAILURE MODE: PROXY LAUNDERING

Do not let a passing test, a linked artifact, or a full board silently become evidence of a broader quality claim than it was designed to support.

Three kinds of statement

An agentic project becomes easier to reason about when it separates three kinds of statement:

1. **A fact about the ledger.** A task changed state, a lease existed, an event was recorded, or a run-case visited a node. These are questions a system can answer exactly if its storage and transition rules are sound.
2. **A fact about a declared check.** A test ran, an artifact was produced, a named gate passed, or a required reference resolves. These can be repeatable, but only prove what the check was designed to observe.
3. **A judgment about the world.** The design is coherent, the product choice is wise, the interaction is clear, or the maintenance burden is acceptable. These may be informed by evidence and review, but they are not made true by a status transition.

The first two categories are where a deterministic coordination system earns its keep. It can make illegal moves difficult or impossible, preserve an audit trail, and display missing evidence without relying on an agent to remember a convention. The third category is where the system must be candid. It can route, record, and preserve the context for judgment; it cannot honestly impersonate the judgment.

Why visibility changes the operating model

The alternative to a lights-out factory is not a dashboard that reports more metrics. It is an operating model in which claims are inspectable. A future reader should be able to ask: what was attempted, which rule admitted the next step, what evidence was attached, which assumption was made, and which questions remain open?

That is the beginning of a glassbox. It does not guarantee a good decision. It makes the decision, its basis, and its limits visible enough to challenge.

The next chapters turn this stance into a practical contract: what the system can enforce, what it records as a claim, and how an agent should respond when it cannot establish the claim required to proceed.

‡ What the reader should keep

1. **A passing test can be valuable and insufficient at the same time.** Its value comes from its stated predicate, not from a general aura of completion.
2. **The first act of control is narrowing the claim.** Name what is established before asking whether the whole product question is answered.
3. **Proxies need a challenge path.** A board, task, or receipt becomes safer when a later reader can inspect the claim it was allowed to support.
4. **Human judgment is not an automation failure.** It is the remaining work when the project lacks an adequate mechanical oracle.

REFLECTION

Which green signal in your current delivery system is routinely asked to support a broader claim than its oracle actually tests?

The Glassbox Contract

Claim status: doctrine grounded in the stable public model. Exact command contracts belong in the verified recipes appendix, not in this chapter.

? The question

What makes a delivery system inspectable rather than merely well logged?

💡 The thesis

A glassbox declares the space under review, makes narrow mechanical rules bite, preserves the provenance behind each claim, and turns uncertainty into a next action. Visibility without those four obligations is observation without a control contract.

A glassbox is not defined by putting every action into a log. Logs can be noisy, partial, and impossible to interpret. It is defined by a contract between the people and agents doing work and the system that records it.

The contract has four parts.

☑ THE GLASSBOX CONTRACT

Declare the space. Enforce the narrow mechanical rule. Preserve provenance. Keep uncertainty visible and operational.

The contract meets the running case

CHECKOUT RECOVERY starts with one false comfort: its normal payment tests pass. The glassbox contract changes what the team does next. It declares the timeout path, status return, and order history as a bounded space rather than leaving them implicit. It names which receipts will support the claim. It keeps the unresolved question of duplicate payment and customer comprehension visible until someone with authority addresses it.

The contract does not add ceremony to a successful task. It changes the meaning of success: the project may move only as far as its declared rule and evidence allow, while the remainder stays inspectable work rather than an inconvenient silence.

1. Declare the space before claiming progress

Progress is meaningful only against a declared boundary. A project must say what is in view, which quality questions apply, and what counts as a unit of work. Without that boundary, a team can make a single successful assertion and call a vast unexamined area complete.

Lettuce's coverage model expresses this as scopes, units, dimensions, and an applicable grid. The general lesson matters beyond any one tool: a denominator is a promise. It says, "these are the things against which we will judge this claim." A board that counts only items someone remembered to create is not a map; it is a selective diary.

2. Put mechanical rules where they can bite

Instructions are valuable, but they are not controls. A sentence asking an agent to acquire a lease or attach evidence remains a request until a system can refuse the invalid alternative. Mechanical rules should live at the transition where a false claim would otherwise become durable.

This changes the role of workflow. A workflow is not a colored list of stages. It is a set of allowed state changes, with prerequisites that are evaluated at the point of change. A refusal is not friction for its own sake. It is a precise answer: this narrow claim cannot be recorded yet because its declared condition has not been met.

3. Preserve provenance without inflating certainty

Evidence should make a claim inspectable, not decorate it. An artifact, task, test, review note, or external reference can establish a relationship between work and a claim. The record should identify the relationship clearly enough that a later reader can re-run, inspect, or dispute it.

But provenance is not magic. A link proves that something was linked. A passing check proves that the check passed. A signature proves that someone signed. The system should never silently upgrade those facts into "the design is good." The stronger the reader's inference, the more explicit the evidence contract must be.

4. Make uncertainty operational

An honest agent needs more states than success and failure. It needs a way to say that a premise is missing, evidence is stale, a check is too weak, a decision belongs to another

role, or the project map itself needs revision. These are not exceptions to the process. They are the moments that prevent a process from becoming a machine for laundering uncertainty into green.

The practical consequence is simple: when an agent cannot establish a claim, it should not force the state forward. It should leave a useful trace of what was tried, name the missing condition, and create or expose the work needed to resolve it. The next agent then inherits a diagnosis, not a fabricated success.

The human boundary is a design choice

Human judgment is often treated as an embarrassing fallback. In a glassbox it is a designed control point. The goal is not to ask a person to rubber-stamp every mechanical transition. It is to prepare concise, evidence-rich decisions at the few points where the question is genuinely semantic: product direction, system shape, trade-offs, exceptions, and risk acceptance.

This puts the human where judgment has leverage and lets deterministic machinery handle repetition, validation, provenance, and recall. Neither side is made to pretend it can do the other's work.

✚ What the reader should keep

1. **A glassbox starts with a boundary, not a dashboard.** The denominator is a promise about what the team intends to examine.
2. **Rules matter when they can refuse a false state transition.** Advice alone is not a control.
3. **Provenance makes claims challengeable.** It does not make their conclusion automatically true.
4. **Uncertainty must change the next action.** A missing premise, weak check, or stale receipt should become visible work, not an omitted footnote.
5. **Human authority is designed into the flow.** It is reserved for semantic and consequential questions, not used as an after-the-fact rubber stamp.

REFLECTION

Which uncertainty in your present workflow is visible but still fails to change the next action?

The Boundary of Knowing

Claim status: doctrine grounded in shipping assertions, provenance, checks, projections, and human-observable records. The four-level teaching model is an interpretive framework, not a product score.



The question

A project board is green. The task is complete. The linked test report exists. The relevant cell is hardened. What, exactly, may a human now believe?

The wrong answer is “the feature is good.” The right answer begins by separating four kinds of statement that often get compressed into a single word such as done



The thesis

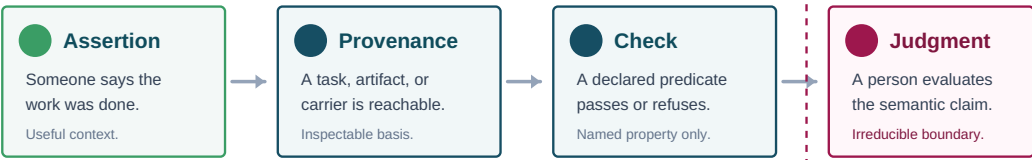
Lettuce is most useful when it preserves the boundary between **assertion**, **provenance**, **mechanical check**, and **human judgment**. Each is valuable. None can impersonate the next.

This is a doctrine of honest operation, not a limitation to apologize for. A system that says less, precisely, is safer to build on than a system that says everything and cannot show why.

Operating principle. Treat every green state as a sentence with an omitted predicate. Ask: green for what claim, by which check, on which evidence, at what time?

One claim, four different kinds of support

Each stage narrows uncertainty. None turns automatically into the next.



Do not infer quality from the green s

The boundary of knowing: assertion, provenance, and mechanical checking support a claim; human judgment remains a distinct semantic decision.

Figure 1. A green state is legible only when the reader can name its claim, evidence, check, and remaining judgment.

Four levels of a claim

1. Assertion

An assertion is someone or something saying that work occurred: “I reviewed the mobile layout,” “this task fixes the race,” or “the migration is safe.” It may be useful context. It is not yet a reason for a system to trust the claim.

2. Provenance

Provenance connects the assertion to inspectable material. A task comment can link to a test report; an artifact can identify a screenshot; a carrier can preserve a candidate patch or decision input; an event can show which action advanced a run-case. Provenance makes a later question possible:

what did this claim rely on?

The presence of a link does not make its destination persuasive. It makes the destination reachable, attributable, and challengeable.

3. Mechanical check

A mechanical check asks a finite, specified question. Did the build compile? Does the artifact reference resolve? Is the expected revision still current? Did the workflow requirement pass? Did a run-case produce the required carrier before advancing? Lettuce can make many such questions bite: a failed predicate refuses the named transition or exposes the gap rather than silently accepting a story.

A check is strong because its scope is narrow. It says nothing outside its predicate. A test suite that passes has checked the behavior encoded in that suite; it has not checked every behavior nobody thought to encode.

4. Human judgment

Human judgment handles the proposition for which there is no adequate mechanical oracle: whether a screen feels coherent across a product, whether an abstraction matches the domain, whether a trade-off is acceptable, whether a customer need was understood. Judgment should be explicit and placed at the decision where it adds the most value. It should not be simulated by a decorative **human** node or by asking a model to announce confidence.

Contract. A deterministic system may refuse a missing declared prerequisite. It may record a human decision and the evidence supplied to that decision. It does not thereby become the author of the judgment.

A case: responsive settings

Consider a settings page that must work on narrow and wide screens without breaking the visual language of nearby pages. The team creates a scope for the settings area, a unit for the page, and a responsive-UX dimension. A task is claimed, the implementation changes, and the agent attaches a screenshot artifact and a test report. The cell cites the task as evidence and its configured guard passes.

This is real progress. Lettuce can now show an address for the work, the task that claimed it, the linked artifacts, the state transition, and the condition under which the cell hardened. If the cited task later fails to resolve, reconciliation can expose the unsupported green. Another reviewer can navigate from the cell to the evidence instead of reconstructing the history from memory.

Now make the overclaim visible. The system has not observed every intended viewport. It has not compared typography and spacing with adjacent screens. It has not tested keyboard focus or an error state. It does not know whether the screenshot was representative. A valid link is not visual review; a passing test is not a design critique.

The proper Definition of Done names the missing judgment: target viewport sizes, states to inspect, comparison surfaces, the evidence to retain, and who is responsible for the final review. A human may perform that review, or a bounded visual regression system may check a subset of it. In either case the claim is stronger because the team named its oracle rather than pretending the task link was one.

Statement after the work	Status	Why
"A task exists and is linked to this cell."	Established provenance.	The store can resolve the reference.
"The configured cell guard passed."	Established mechanical check.	The declared predicate was satisfied.
"The page is coherent at all important widths."	Still a judgment unless a complete, adequate oracle was declared.	The available checks do not cover the full proposition.
"The product is good."	Not a meaningful conclusion from this evidence.	It combines many semantic and contextual questions.

Why the map must be challenged

Cells, units, dimensions, and scopes improve coverage thinking because they make the denominator visible. They do not discover the denominator by themselves. A team can reach 100% hardened cells while missing a whole user journey, operating environment, failure mode, or product constraint that never entered the map.

That is why an honest loop contains discovery and re-opening. A finding can add a unit or dimension, mark a cell as a gap, weaken a stale claim, or send work back to a human decision. Progress is not only the movement of cells toward hardened; it is also the improvement of the map that says which cells matter.

Failure mode: denominator theater. A complete-looking grid can become an excuse to stop looking. A high ratio is evidence about the declared grid, never evidence that the declared grid equals the product.

Three ways a projection can mislead without inventing data

A board is not a neutral window onto stored facts. It is a semantic program: it chooses a denominator, groups states into categories, selects labels, and implies which check is authoritative. The underlying records can be intact while that projection forms the wrong sentence.

Lettuce's own development produced three useful examples. They were open product findings at the 2026-08-02 audit; their value here survives their eventual fixes.

Projection failure	Mechanically available fact	Sentence a reader can wrongly receive	Honest distinction
Empty denominator	The project has zero open cells because it has zero cells.	"Every cell is hardened or excluded."	Unscoped is not complete. Universal truth over an empty set answers the wrong operational question.
Category collapse	Completed, failed, and canceled tasks may all be terminal.	"All milestone work is done."	Closed is not completed. Disposition must survive aggregation.
Checker ambiguity	Validation, diagnosis, and reconciliation can inspect different invariants.	"Strict validation is the strongest whole-store health gate."	A checker's name does not create an authority order. Its contract must say what it checks and what companion gate remains necessary.

These were tracked as **LET-465**, **LET-518**, and **LET-1212**. The lesson is not that dashboards are unreliable by nature. It is that every label is part of the contract. A correct count under the wrong denominator, a correct category under the wrong name, or a correct checker mistaken for a stronger checker can all launder uncertainty without fabricating a byte.

Reader's test. Before accepting a summary, ask what nonexistent items, non-success terminal states, and unchecked invariants disappeared when the projection chose its denominator, category, and authority label.

The practical reading of a green board

When you read a green task, cell, carrier, or graph run, use this sequence:

1. Confirm that the declared set and denominator exist; distinguish unscoped from complete.
2. Name the exact proposition the state and its category represent; distinguish completed from merely terminal.
3. Inspect the evidence, mechanical predicate, and checker contract that supported it.
4. Identify the propositions and integrity classes that checker did not test.
5. Decide whether those propositions are low-risk, need another oracle, or need an explicit human judgment.
6. Record the decision, including uncertainty, so the next reader does not inherit an unexplained green state.

This is how Lettuce becomes a glassbox rather than a dashboard. The visible state is an invitation to inspect the contract behind it, not an instruction to stop thinking.

‡ What the reader should keep

1. **An assertion is not evidence; evidence is not judgment.** Each adds value, but they answer different questions.
2. **A check is only as broad as its predicate.** Its precision is a strength, not a general quality certificate.
3. **A green cell is a claim about a declared coordinate and rule.** It is not a claim that the product has no undiscovered quality work.
4. **Human judgment is designed work.** Name where it belongs, what it reviews, and what evidence it uses instead of treating it as an exception.
5. **Re-opening a claim can be progress.** It may mean the team learned that an old map, artifact, or judgment no longer supports the work.

REFLECTION

For one green cell, can you state separately the assertion, reachable evidence, mechanical predicate, and judgment that remains?

A Map That Can Say Not Yet

Claim status: shipping coverage-map, declared-grid, rollup, and DoD behavior, interpreted through the CHECKOUT RECOVERY teaching case. Choosing a sufficient map remains project judgment.

? The question

How can a project know that it has not inspected work it has not yet written down? A task list cannot answer. It shows claimed items, not the space of work that remains unnamed.

Lettuce addresses part of that problem with a coverage map. The map is not a prediction of every future issue. It is a project-declared statement of which surfaces deserve attention, paired with a mechanism that refuses to let an untouched surface disappear from the denominator. Agents read and maintain that map as they work. Humans observe its declared shape, frontier, evidence, and DoD through the web board and challenge the assumptions it makes visible.

💡 The thesis

A useful coverage map has three parts: a **scope** that partitions a meaningful area of work, **units** that name the things inside it, and **dimensions** that name the quality questions asked of those things. A cell is one coordinate in that map. A Definition of Done sets the bar. The map is valuable only when it can honestly say “not yet.”

This is the first important division of labor in the book. The agent consumes the map operationally: it orients, selects a coordinate, files discoveries, and updates durable state. The human consumes the same map observationally: the board shows what the system currently counts, what remains weak or stale, which bar blocks completion, and where the map itself may be incomplete.

Operating principle. Do not start with a large catalog of quality labels. Start with the decision the project must make, the units that decision covers, and the questions whose absence would make the decision untrustworthy.

Reading a coordinate

A Lettuce cell coordinate is a canonical set of named members, for example:

`scope=checkout;unit=cart;dim=functional`

The names are deliberately ordinary:

- **scope** partitions the board and the Definition of Done. A scope can be a feature area such as checkout, a component area, a release surface, or another bounded concern that the team can review meaningfully.
- **unit** is the item being examined within that scope: a command, screen, service, migration, user journey, or module.
- **dim** is a quality axis applied to the unit: functional behavior, security, accessibility, operability, documentation, or a project-specific question.

The coordinate is not a bureaucracy-shaped filename. It is an address for a question: “has this quality question been answered for this unit in this scope?” Other dimensions may be added where the project needs them. The important part is that each member has a role a reviewer can explain.

Scopes are a decision boundary

`scope=` is reserved because it drives both the board partition and the DoD verdict. A cell without a scope is placed in the synthetic `unscoped` bucket at evaluation time; it is not silently excluded from the project. This protects a team from accidentally creating work that no completion bar sees.

Choose a scope small enough that one status means something and broad enough that its units belong to the same decision. “The product” is rarely a useful scope. “Checkout before the public beta” might be. A scope is not merely a tag; it is the container in which the team will defend a denominator and a bar.

Dimensions are questions, not trophies

A dimension should describe a real angle of evaluation. `functional` may ask whether expected behavior works. `accessibility` may ask whether the interface can be used with the required input and assistive technologies. `security` may ask whether a stated threat or boundary is controlled. Those questions need their own methods and evidence. A cell does not make every dimension applicable to every unit simply because the dimension exists.

This is why Lettuce distinguishes universal and conditional applicability in its dimension model. A project should declare a conditional dimension only where it has a clear reason to apply it. Marking a cell excluded is a decision about the denominator and should carry a rationale, not a quiet way to improve a ratio.

The sparse-map trap

Cells are stored sparsely: a project records the coordinates it has asserted. Sparse storage is efficient, but a list of stored cells is not automatically a coverage denominator. If the only

cells in a project are two hardened ones, a simple count can misleadingly show 100 percent even when the team never named the remaining units and dimensions.

A declared grid repairs that ambiguity. For a scope, the team declares the units and applicable dimensions it intends to cover. The system computes their cross-product. An unstored coordinate in that grid reads as the pack’s default state, normally untested, and remains in the denominator. In plain terms:

$$\text{declared units} \times \text{applicable dimensions} = \text{the questions we promise to answer}$$

If checkout contains **cart**, **payment**, and **receipt**, and the team declares functional plus accessibility as applicable dimensions, the scope begins with six questions. Hardening only the cart’s functional cell is one answer, not a reason to call the scope complete.

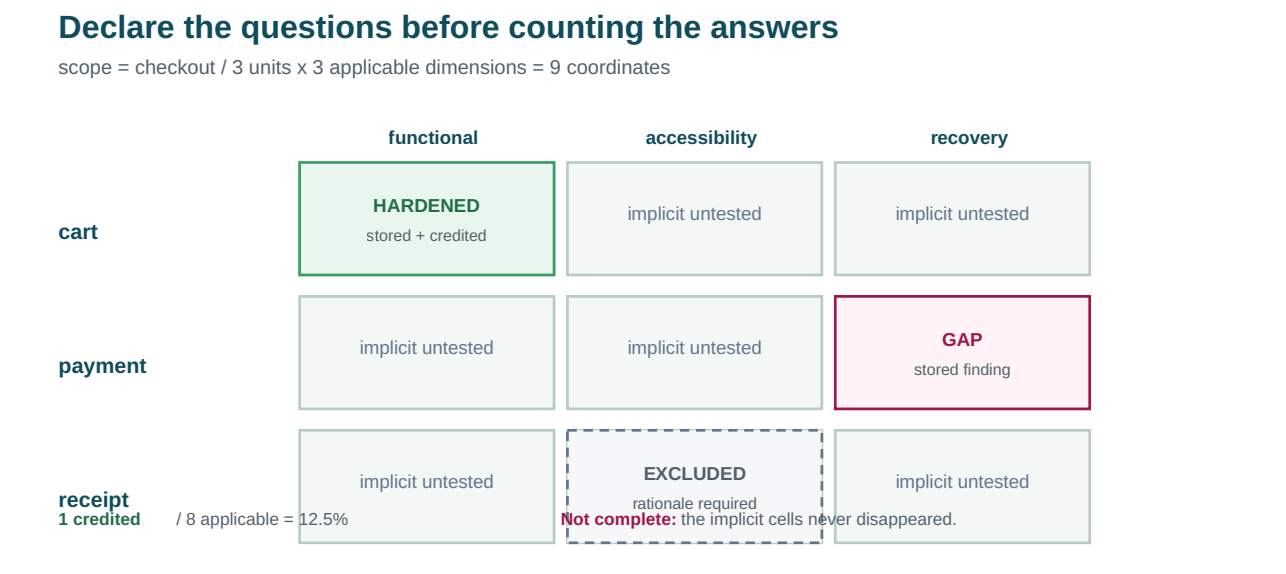


Figure 2. A denominator before a percentage. Shipping model, illustrated with the teaching case: declared units and applicable dimensions create nine questions even when only three coordinates have stored state. The six implicit coordinates remain visible as untested.

Two empty-looking results must also remain distinct. **Zero percent** means the declared population contains applicable work and none of it is hardened. **Not applicable (n/a)** means the population contains no applicable, non-excluded coordinates at all. Treating the latter as zero invents failed work; treating either as complete invents success.

Failure mode: sparse success. A sparse store is not dishonest; it is how the data is stored. The dishonest move is reading a sparse list as if it were the complete population. Declare the grid before using a percentage to make a completion claim.

Definition of Done: a bar, not a mood

The Definition of Done turns the map into a decision rule. It declares a grade floor, and can also require freshness or confirmation depth. The board then reports a computed verdict rather than inviting a human to infer one from a collection of green-looking objects.

This means a scope can be 100 percent hardened by its state ladder and still be DoD-unmet because the required evidence is stale or too shallow. That is not a contradiction. The state answers one question; freshness and depth answer additional questions the team deliberately made part of the bar.

The operating sequence belongs to the agents maintaining the map; the human reviews its declared boundaries and the decision it is being used to support:

1. Define the decision and scope it.
2. Declare the units and the dimensions that genuinely apply.
3. Set the DoD floor and any additional evidence expectations.
4. Read the machine board export and computed DoD verdict, then use **board next** to select the weakest meaningful work frontier.
5. Change the map when discovery reveals that a unit, dimension, or assumption was missing.

board next is guidance, not authority. It can identify the weakest declared area and offer a concrete coordinate to inspect. The rendered web board is an observability surface, not a second evaluator; when a visual meter and the machine export disagree, the exported DoD verdict is the current computational authority and the discrepancy is a product defect to file. Neither surface can tell the team whether the declared map reflects the whole product. That remains the human's periodic review responsibility.

A case: preparing checkout for a beta

The beta decision is clear: a customer must be able to create, pay for, and confirm an order without known release-blocking gaps. The team creates a **checkout** scope. It declares three units - cart, payment, receipt - and three applicable dimensions - functional, accessibility, and recovery. The resulting grid contains nine questions.

At first, the board says not done. The payment recovery cell has no stored work, so it remains an implicit untested coordinate in the denominator. This is useful information, not a failure of the board. It tells the release owner that nobody has yet even recorded how payment timeout recovery will be inspected.

The agent files a task, performs the work, attaches evidence, and attempts the configured guarded hardening transition. That path refuses while its declared conditions are absent. Lettuce also exposes facilitated direct assertions for deliberate recovery and administration; those are not evidence-gated proof and must not be rhetorically merged with the guarded path. During review, they discover that receipt delivery by email is also

part of the beta experience. They add that unit and the grid grows. The percentage may fall, but the release decision has become more honest. The map changed because the product understanding improved.

✚ What the reader should keep

1. **A coordinate is an address for a quality question.** Scope says where, unit says what is examined, and dimension says which question is asked.
2. **A scope carries a decision and a denominator.** It is not a decorative tag.
3. **Sparse storage is not complete coverage.** A declared grid makes untouched but promised work visible as untested.
4. **DoD is a computed bar.** State, freshness, and depth can each matter; one green signal need not settle the decision.
5. **A falling percentage can be good news.** It may reveal a missing unit or dimension before the project turns that omission into a release claim.

REFLECTION

Which release decision does your map support, and what real product surface could remain outside its declared denominator without changing its percentage?

Evidence That Can Age

Claim status: shipping evidence, freshness, confirmation-depth, and DoD behavior. Reviewer diversity and evidence relevance remain human judgments.

The question

What should happen to a claim that was well supported when it was made, but was made before the surrounding system changed? A project that never reconsiders old evidence becomes a museum of green states. A project that rechecks everything on every change becomes unable to move.

Lettuce gives the human operator a middle path: retain the evidence, expose its age and depth, and make those properties part of the Definition of Done only where the decision warrants them.

The thesis

Evidence is not a permanent quality certificate. It is a durable, inspectable relationship between a claim and its basis. Freshness asks whether that basis was bound to a recent store revision; depth distinguishes repeated confirmations, and evidence-backed verification depth requires a distinct evidence reference for that cell. These are useful signals for deciding when to revisit a claim. They are not semantic proof or proof of independent reviewers.

Operating principle. Preserve evidence so a later reader can challenge it. Measure currency and repeated confirmation so the project can decide where to revisit it. Do not confuse either measure with truth.

Three different questions

When a cell is green, three questions are easy to confuse:

Question	Lettuce signal	What it establishes
Was a basis linked?	Evidence reference and its resolution.	A task or external reference is reachable from the cell.
Was the basis recently affirmed?	Freshness bucket.	

Question	Lettuce signal	What it establishes
		The claim has, or lacks, a recent revision anchor.
Was the grade revisited with distinct support?	Confirmation depth, including evidence-backed verification depth.	Separate revision confirmations were recorded; strong verification credit also used a distinct evidence reference for that cell.

Each question is narrower than “is this still good?” A task reference can resolve while its test no longer covers the changed behavior. A fresh affirmation can still be shallow. Multiple confirmations can be correlated and miss the same design error. The signals are useful because they show this structure instead of hiding it behind a single green badge.

One claim, two different reasons to revisit it

Freshness measures store movement. Depth records corroboration structure.

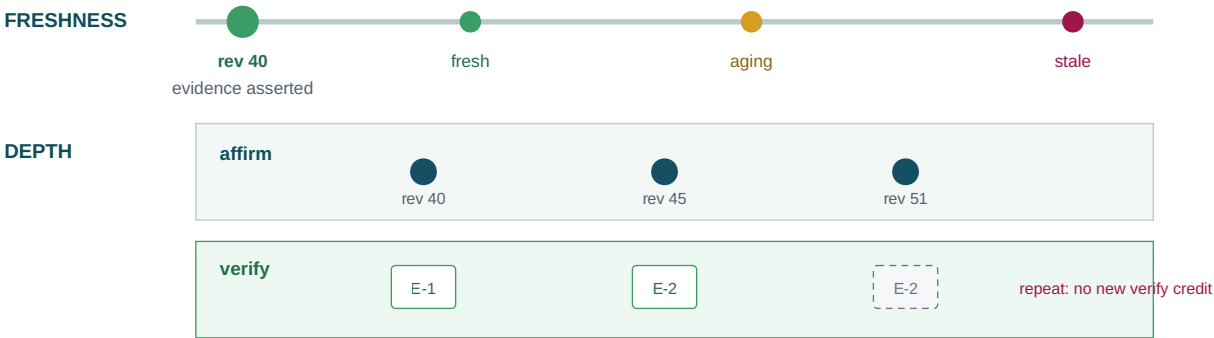


Figure 3. Currency and corroboration are separate axes. Shipping model: freshness derives from distance between store revisions; depth records confirmations. Evidence-backed verification depth additionally requires a distinct evidence reference for the same cell.

Freshness is movement, not the clock

Lettuce computes freshness from store revisions rather than elapsed wall time. The distance is the current store revision minus the revision where the cell’s evidence was last asserted. The result is a derived bucket: **fresh**, **aging**, or **stale**. A cell with no real revision anchor is marked **presumed**, an honest sign that it has never bound its currency to a recorded evidence assertion.

This choice matters. A quiet project should not invalidate a careful review just because a calendar page turned. A rapidly changing project should not call old evidence current merely because it is only a few days old. Revision distance is still an imperfect proxy for relevance, but it is at least tied to recorded work movement and recomputed on read rather than stored as a value that can drift.

Depth is not repetition

Depth counts confirmations at distinct store revisions. Repeating an affirmation at the same revision does not buy depth. A verification earns strong `depth_verify` credit only when it carries an evidence reference not previously used to verify that same cell; reusing the reference can still record an affirmation but does not buy another strong verification. The same evidence may support a different cell, because distinctness is evaluated per cell. These rules resist one simple rubber stamp without claiming to detect every one.

Depth does not establish reviewer independence. Two agents using the same prompt, model, repository snapshot, and weak test can produce two confirmations of the same blind spot. The probability chapter explains why that correlation matters. Depth records separate confirmation events and, at the stronger tier, distinct per-cell evidence references. The human still decides whether the reviewers, methods, and evidence are meaningfully diverse.

DoD makes the bar explicit

The Definition of Done can require a minimum grade, a recency floor, and a confirmation depth. It evaluates these as a strict AND gate per scope: every applicable, non-excluded cell must clear every declared floor. The project can also remain not done because milestones are incomplete or non-terminal tasks remain.

This produces a valuable result: a scope may be 100 percent hardened and still be DoD-unmet. That is not a defect in the board. It tells the reader that the state ladder was cleared but the decision demanded fresher or deeper support.

Contract. A DoD is a declared measurement rule. It can refuse the completion claim represented by its floors. It cannot decide whether those floors are sufficient for the product, customer, or risk at hand.

A case: checkout recovery after a change wave

The payment-recovery cell in CHECKOUT RECOVERY reached `hardened` after a narrow-width review, timeout tests, and linked task evidence. The team then changes payment-state handling and order-history integration. The original evidence still resolves, but its revision anchor recedes as store events accumulate. The cell becomes aging, then stale under the project rule.

The right response is not to declare the old review false. It was a valid fact about the work at that time. The right response is to reopen the question: does the old review still support the current system? The team performs a focused review, adds new evidence, and records a

new confirmation. If it discovers a gap, it changes the cell and files work rather than preserving a historical green as a present-tense claim.

The human must still decide the review's scope. Revision distance cannot know whether a copy-only change affects recovery meaning, just as it cannot know whether a small payment-state change has large consequences. It is a prompt to inspect, not an autonomous verdict about relevance.

‡ What the reader should keep

1. **Evidence is durable provenance, not permanent proof.** It lets a later reader find and challenge the basis of a claim.
2. **Freshness measures recorded movement.** It is a reason to re-evaluate, not a calendar-based accusation that old work was bad.
3. **Depth resists specific forms of repetition.** Distinct revision confirmations matter; strong verification also needs distinct per-cell evidence. Neither signal establishes independent judgment.
4. **DoD can be stricter than the coverage ratio.** A hardened state can remain insufficient when the decision requires fresher or deeper support.
5. **Reconsideration preserves honesty.** A stale claim should become visible work, a new review, or an explicit human decision - never invisible debt.

REFLECTION

What change could make a one-revision-old receipt irrelevant while leaving a hundred-revision-old receipt valid, and who is accountable for making that distinction?

The Loop That Remembers

Claim status: shipping task, lease, workflow, board, artifact, cell, and evented-record mechanisms, assembled here as an operating doctrine. The exact loop policy belongs to each project.

? The question

How can a human observe several agents, tasks, reviews, and discoveries without receiving only a stream of plausible but disconnected activity? The answer is not to ask every agent to remember more or to make the human operate every transition. It is to give the agents one durable loop whose reads orient the next action and whose writes become evidence for both the next agent and the human observer.

💡 The thesis

Lettuce's agentic loop is an agent-operated, human-observable discipline:

`orient -> act -> reflect -> file -> orient`

Agents execute the loop and maintain its map and ledger. Humans observe its state, govern consequential authority, challenge quality bars, and decide at declared boundaries. The store is not an agent's mind. It is the project's inspectable memory.

The worker may vanish; the loop must remain

Each move reads or changes durable state before another agent inherits the work.

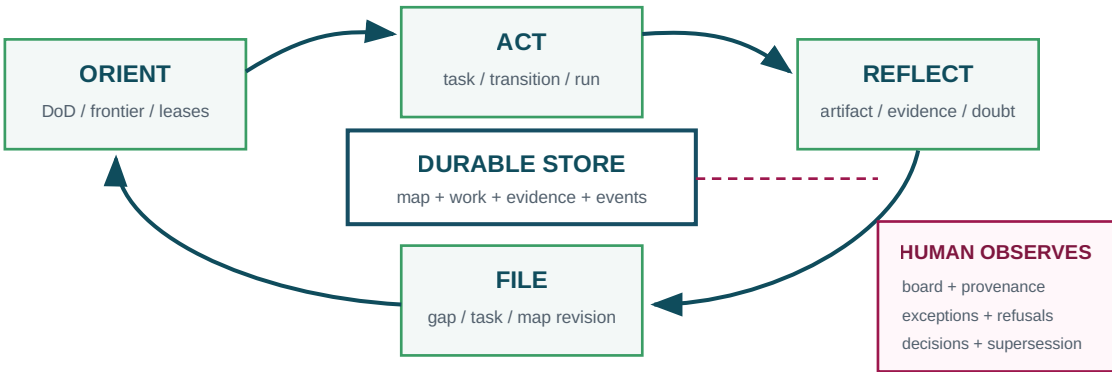


Figure 4. A loop that survives its worker. Doctrine assembled from shipping records: agents repeatedly read and mutate durable project state; humans observe and govern the resulting state rather than depending on an agent's private context.

Operating principle. Never let an agent’s private context be the only place where the reason for work, the evidence for a claim, or the discovery of a gap exists.

A factory needs a supervisor, not only a ledger

The four-phase loop is necessary, but it is not yet a software factory. A ledger can remember actions without selecting them well. A board can expose a frontier without dispatching a capable worker. A graph can preserve a route without operating the model named in its execution policy. A human node can appear in a definition without an authenticated human decision ever arriving.

The stricter model is a supervised feedback system. Autonomic-computing work described a monitor-analyze-plan-execute loop over shared knowledge; production reliability practice adds the less glamorous requirement that the loop’s own latency, errors, traffic, and saturation remain observable (IBM, [On Autonomic Computing Architectures](#); Google SRE, [Monitoring Distributed Systems](#)). The analogy is useful only if every part is named:

Factory element	Concrete software meaning	What Lettuce supplies today	What remains outside or incomplete
Reference	Goal, declared scope, DoD, workflow, risk and effect policy.	Durable scopes, dimensions, milestones, DoD, workflows, graph definitions.	Whether the reference describes the right product; who may revise it.
Controller	Selects work, chooses a worker, sets a budget, reacts to outcomes.	A structured frontier and graph/run state for a controller to read.	Central scheduling, capability matching, priority aging, load balancing, and adaptive cadence.
Actuator	An agent and its tools change code, tests, configuration, or external systems.	Attributed tasks, leases, runs, transitions, carriers, and effect assertions.	The actual tool sandbox, credentials, rollback, and external-effect authorization.
Plant	Repository, build, service, users, and operating environment being changed.	References and artifacts about the plant.	The plant itself and independent truth about its outcomes.
Sensors	Tests, screenshots, traces, reviews, incidents, cost, and downstream outcomes.	Durable artifacts, evidence links, telemetry, route facts, and events.	Proof that a claimed observation occurred or was adequate for its conclusion.
Estimator	Turns observations into current state and a next-work frontier.	Board export, DoD, freshness/depth, queries, reconciliation, replay.	A bounded-cost orientation path at canonical-store scale; a guarantee that the map covers reality.

Factory element	Concrete software meaning	What Lettuce supplies today	What remains outside or incomplete
Safety supervisor	Allows, holds, refuses, escalates, suspends, or revokes authority.	Selected gates, workflow requirements, deterministic graph refusals, leases, and human-shaped nodes.	Authenticated decision receipts, enforced graph staffing policy, evaluator qualification, and independent revocation.
Knowledge	Durable state from which the next cycle and later audit can recover.	The filesystem store and its versioned/evented records.	Tamper resistance against an actor with write access to the same trust domain.

The ledger is memory; the factory still needs a supervisor

A controlled shift closes the loop from policy to action to evidence to the next authorized move.

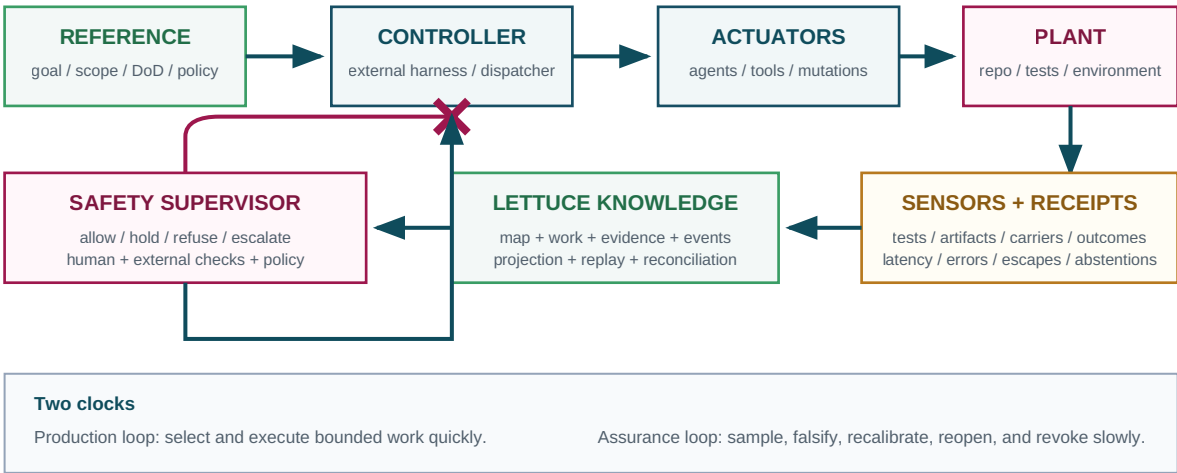


Figure 5. The factory control plane. Mixed status. Lettuce supplies much of the knowledge, projection, and narrow refusal layer. The dispatcher, actual execution environment, independent sensors, authorization boundary, and semantic supervisor remain external or proposed. Closing the arrows in a diagram does not close them in production.

This decomposition prevents a recurrent category error: calling durable state an autonomous controller. Lettuce is closest to the knowledge and control-plane substrate. An external harness still decides when to wake, which candidate to choose, which agent and model to dispatch, which tools and credentials it may use, how long it may run, and what to do when it stalls. The current public orchestrator guide explicitly leaves central scheduling, capability matching, automatic load balancing, priority aging, and notifications for later. Graph `exec_policy` is authored, linted, displayed, and hashed staffing intent; the run-case engine does not execute it. Graph enactment is not yet an HTTP surface.

That boundary does not make the existing loop useless. It makes the integration contract visible. A practical factory can use Lettuce as its durable control state while keeping dispatch, tool execution, secrets, and external-effect policy in a harness designed for those responsibilities.

Two loops run at different clocks

A production loop selects and executes bounded work. An assurance loop asks whether the production loop is still producing the outcomes it was designed to produce. Compressing both into one fast loop is dangerous: the same agent that writes the work can then grade the work, revise the bar, and authorize the next effect before any independent signal arrives.

The production clock may be minutes: orient, claim, execute, verify a narrow contract, record, and select again. The assurance clock may be hours, releases, or outcome cohorts: sample full traces, inject faults, compare projections, inspect escaped defects, recalibrate proxies, reopen claims, and reduce or revoke authority. Some delayed outcomes cannot be accelerated merely because the factory wants an answer.

Capability benchmarks reinforce the distinction. METR's time-horizon work measures the length of well-specified tasks agents can complete with a chosen success probability, while its randomized study of experienced maintainers showed that benchmark capability did not automatically translate into higher productivity on familiar repositories (Kwa et al., 2025; METR, 2025 developer-productivity study). These results are snapshots, not permanent limits. Their durable lesson is that task completion, throughput, and factory value are different measurements.

Orient: read before acting

board next is an agent's operational starting point. It reads the current board, leads with the Definition of Done verdict when one is declared, ranks scopes weakest-first, and offers a concrete next coordinate at greater detail levels. It is a read, not an order. An agent uses it to avoid inventing a target from stale context. A human sees the same declared frontier through the rendered web board and can inspect why the system reports progress, weakness, staleness, or blocked DoD.

The agent treats the machine export and DoD verdict as the structured projection. The rendered board supports human scanning and challenge, but it is neither a separate authority nor guaranteed to be cheap on a very large event store. A slow or internally inconsistent projection is itself a finding to file, not a reason to plan from memory.

Before allowing new work, inspect the DoD verdict, the open frontier, and work already in flight. A valid plan may say "do not start": the relevant task might already be leased, the scope may be blocked by stale evidence, or the map itself may need revision.

Orientation is also a control-plane heartbeat and therefore needs a latency budget. On Lettuce's canonical store, **LET-1415** measured the normal board path walking the full event history. **LET-1417** reduced three traversals to one and preserved the answer, but measured export time remained 287-288 seconds under the recorded conditions. The important conclusion is not that a particular number will hold on every machine. It is that an autonomous loop whose cheapest orientation read grows with all historical events

eventually loses its clock. Agents time out, fall back to private plans, or start duplicate work precisely when the durable store becomes most valuable.

Act: claim an accountable unit of work

A task makes an intention addressable. A lease makes exclusive work explicit and expiring. A workflow transition records that the task entered a named state under its declared requirements. These are coordination facts, not proof the task is valuable, but they prevent silent duplicate effort and unowned mutations.

For an agent, permissions should match the action. Exploration can be read-only; a reversible change can be constrained to a task and branch; a consequential transition can require a lease, evidence, expected revision, or human approval. The point is not ceremony. It is making the authority to change shared state as visible as the change itself.

A lease prevents two workers from legitimately claiming the same task. It does not prove that two differently named tasks are semantically independent, that their patches do not overlap, or that the store can sustain an arbitrary number of writers. Factory concurrency therefore needs a bounded work-in-progress policy above the lease mechanism: how many workers may run, which resources they share, how conflicting patches are detected, and when dispatch stops because review or integration is saturated. More agents can increase queueing and correlated rework even while every lease is valid.

Reflect: turn work into inspectable evidence

After acting, record what occurred: an artifact, a run log, a comment, a task transition, a carrier, or a cell evidence reference. Then ask the adversarial question: what did this work fail to establish? A passing test can support a behavioral claim. It cannot by itself settle design fit, UX coherence, or every unmodelled failure.

Where a cell is meant to harden, the evidence and its configured guard should actually bite. If the guard refuses, the refusal is a useful result: it tells the next loop what condition remains missing. Do not bypass it to make a dashboard look healthy.

File: make discovery part of the plan

The most important loop step is often the least glamorous. If work discovers a defect, create a task and mark the coordinate as a gap. If it discovers an untested surface, add it to the map or declared grid. If it invalidates evidence, reconcile or re-open the claim. If it changes the release hypothesis, update the milestone or escalate the decision.

Filing is how the project prevents a finding from dying with the agent session. It changes the next **orient** result. Work hardens cells; cells reveal work.

A viable autonomous shift today

An imperfect but honest shift can run now if the external harness treats Lettuce as control state rather than as the whole controller. One bounded iteration is:

1. **Preflight the control plane.** Confirm store health, projection latency, writer availability, credential scope, and that the previous shift left no unresolved recovery state.
2. **Orient from durable state.** Read the DoD, frontier, in-flight work, milestones, and known exceptions. If orientation exceeds its budget, stop; do not silently plan from memory.
3. **Select under policy.** Prefer consequential gaps over easy green counts, respect WIP and dependency limits, and name why this work outranks another candidate.
4. **Claim and budget.** Acquire the task lease, pin the expected revision, choose a worker with appropriate capability, and bound time, cost, tools, filesystem access, network access, and external effects.
5. **Execute in a reversible lane.** Make the change and preserve intermediate findings. A timeout, refusal, or abstention is an outcome, not permission to weaken the bar.
6. **Verify the narrow claim.** Run the declared checks, including a biting negative control where the claim needs one. Bind receipts to the subject revision and tested scope.
7. **Separate production from approval.** Do not let the worker's own statement, a duplicated branch, or a second prompt in the same context masquerade as independent verification.
8. **Commit the record.** Attach artifacts and evidence, transition only through satisfied gates, record remaining doubt, and preserve facilitated or waived state as exceptional rather than earned green.
9. **Reconcile and widen.** Detect derived-state drift, file every unresolved finding, add missing map surface, and reopen invalidated claims.
10. **Decide whether another iteration is authorized.** Continue only while the control plane is healthy, the autonomy budget remains inside its bounds, and the human exception queue is not accumulating beyond policy.

The shift stops on store invalidity, an unresolvable conflict, stale or missing evidence at a required boundary, exhausted loop/cost/time budget, an unmodeled irreversible effect, missing authority, repeated no-progress outcomes, or an orientation path that cannot return within budget. "Keep trying" is not a recovery policy.

Measure the factory, not its motion

Tasks closed and cells hardened are output counters. A factory needs control signals that can reveal when output is becoming less trustworthy:

Signal	Why it matters	Dangerous interpretation
Orientation latency and timeout rate	Whether agents can still enter the shared loop.	Treating timeout as permission to use a private plan.
False-green and reopen rate	How often accepted work later loses its claim.	Treating reopening as failure and suppressing it.
Correction latency	Time from finding to changed map, task, evidence, or decision.	Counting only detection throughput.
Independent reconstruction success	Whether a new reader can recover reason, evidence, authority, and doubt.	Equating event volume with auditability.
Human queue age and review minutes	Whether autonomy has displaced work into an invisible bottleneck.	Calling low interruption count efficiency while decisions wait.
Lease conflict, patch conflict, and abandoned-run rate	Whether parallelism is useful or merely concurrent.	Assuming valid leases imply composable work.
Escapes and delayed outcomes by decision lineage	Whether the assurance loop reaches consequences.	Optimizing short-window pass rates.
Facilitation, waiver, and override rate	Whether exceptions are becoming the normal route.	Folding exceptional green into earned coverage.

Google’s monitoring guidance emphasizes latency, errors, traffic, and saturation because a service can be busy and unhealthy at the same time. The same is true of an agent factory. Throughput without correction, saturation, and escape signals is motion without control.

A case: an agent finds the wrong success path

An agent takes the board’s weakest checkout coordinate, claims a task, and runs the existing tests. The tests pass, but manual inspection shows that a payment timeout leaves the customer in an ambiguous state. The agent must not complete the cell as if the passing suite were the whole story.

It records the test artifact, files a task for the timeout behavior, marks the cell as a gap with a reason, and adds the recovery surface to the checkout map if it was absent. The next **board next** now reveals the new work instead of steering another agent toward a false green. The loop improved its own model of the project.

Failure mode: context evaporation. An agent can reason well, discover a real problem, and still make the project worse if its discovery stays only in a transient conversation. The remedy is not a longer prompt. It is a durable task, cell, artifact, or decision record that changes the next read.

‡ What the reader should keep

1. **A ledger is not a supervisor.** Lettuce preserves shared control state; an external harness still chooses work, enforces budgets, executes effects, and responds to stops.
2. **Production and assurance run at different clocks.** Fast work selection cannot replace slower revalidation, reopening, and outcome review.
3. **Orientation is a bounded safety step.** An agent must recover the DoD, frontier, in-flight work, and authority before acting, and stop if that read cannot complete within policy.
4. **An autonomous shift needs explicit bounds.** WIP, time, cost, loop, privilege, effect, and exception budgets turn “keep going” into a controlled operating interval.
5. **Measure correction, not only motion.** Reopens, reconstruction success, correction latency, escapes, and human-queue age reveal health that task throughput cannot.

🕒 REFLECTION

If the current agent vanished after its next action, which durable record would let a different agent recover the reason, evidence, remaining doubt, next decision, and reason the shift was still authorized to continue?

Method, Evidence, and Enforcement

Claim status: shipping methodology fields, evidence, guards, freshness, depth, reconciliation, workflow requirements, and graph refusals, combined with doctrine about their honest use. Semantic inspection quality is not a mechanically verified product claim.

The question

When a dimension says “inspect every screen,” how does a project know whether an agent actually did that work? More importantly, how does it avoid accepting a plausible artifact as proof that the inspection was careful or that the product is coherent?

The thesis

A reliable operating system separates three different things that are often collapsed into one word,
done
:

Layer	What it supplies	What it cannot honestly supply
Methodology	A procedure for directing attention and making a judgment.	Proof that the procedure was followed well.
Evidence contract	A declared receipt a system can check for presence, shape, freshness, or linkage.	Proof that the receipt supports the intended conclusion.
Judgment	An accountable evaluation of context, tradeoffs, and consequences.	Mechanical repeatability or universal agreement.

Lettuce has useful parts of the first two layers. A dimension can carry a methodology that tells a worker how to apply the axis to a unit. Cells, evidence, guards, freshness, depth, workflow requirements, and graph effects can make selected receipts durable and can refuse selected state changes. The third layer remains irreducibly accountable human work when the claim is semantic: whether an interaction makes sense, whether an architecture is adequate, or whether a tradeoff is acceptable now.

This is not a weakness to hide. It is the boundary that prevents a project from mistaking procedural compliance for quality.

A methodology is an instruction to look

A dimension is not useful merely because it has a reassuring name such as “accessibility” or “visual coherence.” Its methodology says what a serious application of the dimension entails. For a responsive-interface dimension, a methodology might require the worker to:

1. inspect each declared screen at the relevant viewport classes;
2. follow the important transition into and out of the screen;
3. compare related screens for repeated controls, labels, and spacing;
4. record exceptions and the observation that led to them; and
5. decide whether the result reaches the dimension’s stated bar.

That procedure does two jobs. It gives an agent a less ambiguous work plan, and it gives a reviewer something concrete to challenge. A reviewer can ask whether the checkout timeout was included, whether the narrow viewport was tested, and whether the comparison set was real. Without a methodology, the grade becomes a personal adjective attached to a coordinate.

But a methodology is not an execution trace. An agent can repeat its language without opening a browser. It can take screenshots without looking at the transition. It can report a comparison without noticing that the same destructive button changes meaning on a neighboring screen. A detailed procedure makes bad work easier to detect; it does not make bad work impossible.

Method is a promise of attention. It should make an agent’s procedure and a reviewer’s challenge explicit. It should never be presented as evidence that the attention occurred.

An evidence contract is a narrow refusal rule

An evidence contract names what the system can actually test before accepting a claim. It may require a linked artifact, a test result, a current evidence item, a named transition, a sufficiently deep confirmation, or a guard whose condition has been satisfied. The useful word is

require

: when a declared prerequisite is missing, the system can refuse the relevant hardening or transition rather than quietly accepting a green label.

This is valuable precisely because it is narrow. A system can check that the release candidate has a screenshot artifact. It can check that a required test receipt belongs to the work. It can check that an old receipt has aged beyond a configured freshness bar. It can check that a gate has not received its declared input. Those are structural and mechanical facts.

It cannot infer that a screenshot reveals every confusing layout, that a test meaningfully exercises the failure path, or that a reviewer understood the customer consequence. A receipt can be genuine and still be weak evidence. A gate can be satisfied and still be aimed at the wrong question.

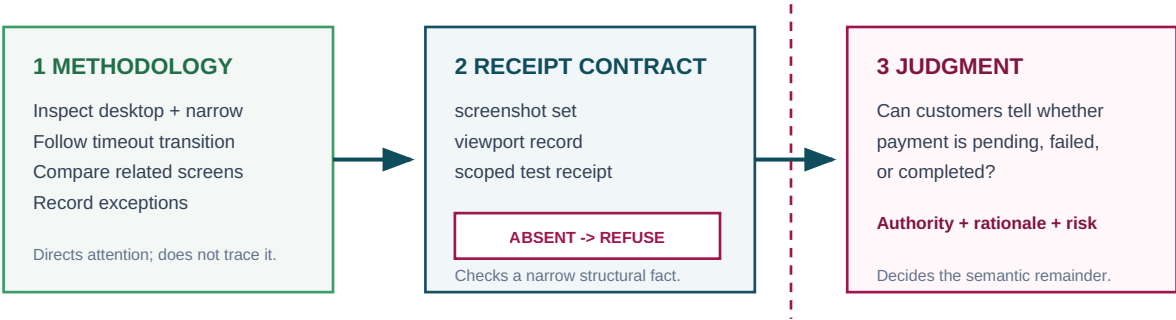
The design discipline is therefore to make each contract state its exact claim:

Require: screenshot + viewport record + test receipt
For: the claim that the checkout timeout was inspected at the declared widths
Not for: the claim that the recovery experience is good enough to release

The first statement is a candidate for enforcement. The second must remain open to judgment and challenge.

Three layers answer three different questions

Direction can be explicit. Receipts can bite. Meaning still has an owner.



A receipt crossing layer 2 is not an answer to layer 3.
The system can expose the remainder and route it to authority; it should not counterfeit the decision.

Figure 6. Direction, refusal, and judgment. Mixed status: Lettuce ships methods, receipts, selected guards, and refusals; the broader three-layer operating model is doctrine. A receipt can cross the mechanical boundary without settling the semantic one.

Enforcement also has a lifecycle. A guard can refuse a gated transition when a required receipt is absent. If the last evidence supporting an already-hardened cell is later removed, current rollups and boards stop crediting that hardened state; reconciliation then demotes the stored assertion. The system preserves the contradiction long enough to expose and repair it rather than continuing to count a hardened lie. This standing check is narrower than a universal invariant: a deliberately facilitated direct assertion that never had evidence belongs to a separate population and is not retroactively converted into proof.

Enforcement strength comes from trust separation

Counting gates is not the same as counting independent controls. Identity, leases, graph events, cell evidence, DoD, and task transitions can all be useful against omission and coordination error while still living in one trust domain: the same agent may be able to author the work, the receipt, the grade, and the policy file. Five agreeing records written by one authority remain one authority.

The open adversarial review [LET-719](#) states the uncomfortable hypothesis directly: most in-store controls may be self-attestation against accidental failure, while an external protected check is the only layer the worker cannot rewrite. Its graph dogfooding exposed the same issue in miniature. A quorum can prove that three branches arrived; it cannot prove that their reasoning, evidence, authors, models, or failure modes were independent. The runtime enforces shape. Independence requires separately observable lineage and an authority boundary.

Enforcement level	What it means	Typical factory value	What must not be inferred
Advisory	A methodology, instruction, or execution policy tells an agent what it should do.	Repeatable intent and a basis for review.	That the instruction was followed.
Recorded	The ledger preserves an attributed claim, receipt, route, or exception.	Recovery, comparison, and later challenge.	That the record is true or independently witnessed.
Gated	A deterministic predicate refuses a state change when its declared input is absent or invalid.	Protection against omission and malformed transitions.	That the predicate is sufficient or outside the worker's control.
Independently controlled	A different trust domain owns the check, credential, effect, or revocation.	Resistance to self-approval and motivated bypass.	Universal correctness beyond that control's oracle.

This ladder is not a ranking in which recorded evidence is weak and therefore pointless. Each level answers a different threat. The error is presenting one level as another. A facilitated hardening event is still valuable as an honest record of imported or hand-graded state, but its failed gate verdict must remain queryable and excluded from any denominator claiming earned proof. An external CI check can independently enforce a test command, but it still cannot decide whether the test expresses the right product behavior.

NIST's work on evaluation probes uses the same separation: probes operate inside an agentic workflow, compare claims with a curated reference corpus, and accumulate a machine-readable audit trail. The useful advance is an observable challenge path, not a declaration that the evaluator became an oracle ([NIST, Building Evaluation Probes into Agentic AI](#)).

A case: the responsive checkout that passes its receipts

Consider a checkout scope with a responsive-review dimension. Its methodology asks the worker to inspect desktop and narrow widths, the card-decline path, and the payment-timeout path. The evidence contract requires a screenshot set, a test report, and a short review artifact before the cell can harden.

An agent produces all three. The tests pass. The screenshots exist at both widths. The review artifact says that the checkout is coherent. Structurally, the work is stronger than an unsupported assertion: the project can see what was submitted and can refuse a missing receipt.

A human reviewer opens the timeout screenshot and notices that the customer is returned to the basket without knowing whether the card was charged. The artifacts were not fabricated. The methodology was not necessarily ignored. The problem is that the final semantic question was never reduced to a valid mechanical predicate.

The honest response is not to delete the evidence or declare the entire system useless. The reviewer records the concern, opens or updates the task, regresses the cell or leaves it unhardened as the current contract permits, and makes the release decision explicit. The next methodology revision might add a question: “Can a customer tell whether payment is pending, failed, or completed?” The next evidence contract might require a state-transition capture. Neither change makes the product automatically good, but both make the next pass less likely to miss the same ambiguity.

What an agent loop must do

For an autonomous agent, this separation becomes a concrete rhythm:

1. **Read the methodology before grading.** It names the work to attempt, not merely the label to assign.
2. **Declare the receipts before acting.** Know which artifacts, checks, links, or gate inputs the claim will require.
3. **Perform and record the work.** Attach evidence that another reader can inspect rather than a private conclusion.
4. **Test the contract mechanically.** Let a missing or stale receipt refuse the state change; do not explain it away.
5. **Challenge the semantic remainder.** Ask what the receipts cannot establish, and escalate that question when the actor lacks authority to decide it.
6. **Improve the method after a miss.** A discovered blind spot is a change to the future procedure, evidence bar, scope, or decision boundary.

Before citing a test receipt across concurrent work, bind it to both the subject revision and the scope actually exercised. A base revision alone says where the test began; it does not

say which commands, packages, screens, or paths the receipt covered. Without both coordinates, a later merge can make the evidence look current while leaving its relevance uncheckable.

This is enforcement of behavior in the only honest sense available to a deterministic system: it can constrain the shared state transition around a declared contract. It cannot reach into an agent's private process and certify that the agent looked, understood, or cared.

Control theater and useful friction

Projects often respond to unreliable agent work by adding more fields, more checklists, and more green labels. That can create control theater: workers learn to supply receipts that satisfy the form while the underlying question remains unexamined. The cure is not to remove every contract. It is to make each one pay for itself.

A useful enforcement rule has a named failure it can catch, an artifact that is cheap enough to inspect, a clear owner, and a recovery path when it refuses. For example, requiring a fresh visual artifact can catch a cell hardened against an obsolete interface. It does not need to pretend that pixels decide product fit. The resulting friction directs attention where a silent failure would be costly.

Failure mode: receipt laundering. A passing check, linked task, or completed checklist becomes dangerous when it is cited as proof of a broader claim than it was designed to support. Preserve the exact proposition beside the receipt, and keep the semantic remainder visible.

‡ What the reader should keep

1. **Methodology directs attention.** It makes a dimension gradeable and challengeable; it does not prove the work happened well.
2. **Enforcement is narrow by design.** It can refuse a missing declared prerequisite or malformed receipt, not certify product quality.
3. **Evidence is not its conclusion.** A real artifact can still be irrelevant, stale, incomplete, or interpreted badly.
4. **Control strength comes from trust separation.** A second branch, prompt, or signature is not independent review when the same actor controls both production and approval.
5. **Authority belongs at the semantic remainder.** Name who may decide it, preserve dissent, and use every miss to revise the method, receipt, map, or decision boundary.

REFLECTION

Which part of your quality bar can fail closed today, which part merely records a receipt, which actor controls each, and which proposition still requires an accountable person to decide?

Graphs Are Contracts Before They Are Pictures

Claim status: shipping graph-definition and run-case behavior, interpreted through the CHECKOUT RECOVERY teaching case. The release decision in the case remains human judgment, not graph output.

The question

What does a graph add after a project already has tasks, evidence, and a human release owner? It adds a durable answer to a different question: which bounded steps may happen in which order, with which declared handoffs, and what should refuse when that contract is not met?

The thesis

A Lettuce graph is not an automation drawing. A graph definition is a versioned, soundness-checked coordination contract; a run-case is one evented enactment of that contract. The distinction matters because a node's shape does not contain the behavior of its worker, and no topology can make a semantic release decision true.

For CHECKOUT RECOVERY, the graph is useful when it expresses a narrow review route: collect the timeout evidence, inspect it against a named question, and present the unresolved risk to the release owner. It is not useful when it pretends that a diamond, a quorum, or a node labelled **human** has decided whether customers are safe.

Three layers, three questions

Graph definitions keep three different ideas separate:

Layer	Question it answers	Example in <u>CHECKOUT RECOVERY</u>
Concern	What structural role is this station?	A producer, reviewer, router, gate, human, or terminal.
Instruction	What should the worker at that station do?	Compare timeout wording with order-history state; list unresolved customer interpretations.
Execution policy	How is an orchestrator intended to staff an agent station?	

Layer	Question it answers	Example in CHECKOUT RECOVERY
		A declared model/capability token, effort level, and resident or ephemeral context.

This separation is a defense against an attractive but empty diagram. A node called **reviewer** describes a role, not a review method. A node called **human** draws a human-shaped structural station, not an authority model, a meeting, or a legally meaningful approval. An instruction gives a worker a behavioral contract. An execution policy records how an orchestrator is intended to staff an agent station. It changes the compiled identity and appears in authoring and visualization, but Lettuce's run-case runtime does not currently consume or enforce its model, effort, or residency fields. The orchestrator remains part of that control boundary. Each layer can be inspected and revised without pretending it is the other.

The current authoring surface stores these details in a versioned graph definition and computes an effective hash over the expanded contract. A changed instruction, route guard, or execution policy therefore changes the identity of what is being enacted. That is a useful provenance fact: later readers can ask which coordination contract a run followed. It is not evidence that the contract was a good one.

The smallest useful checkout graph

READ THE RUNTIME SEAM FIRST.

The definition declares the contract, but current run-case advancement still receives some carrier, quorum, loop, and route terms from its caller. A definition can therefore be sound and versioned while a weak orchestrator enacts a weaker supplied contract. The runtime refuses the terms it is given; it does not yet make every definition declaration authoritative at advance time. Chapter 9 follows this boundary in detail.

The running case begins with four stations:

`evidence packet -> timeout review -> release question -> terminal`

The evidence packet station produces the screenshot set, test receipt, and observed timeout state. The reviewer receives a narrow instruction: compare the timeout message, retry affordance, and order-history result; identify whether a customer can distinguish pending, failed, and completed payment. The release question then belongs to the accountable owner. The terminal records that this bounded review path ended; it does not claim the product was approved.

The graph's value is that each handoff becomes visible. A later reader can see which station produced which carrier, what route was taken, and which effect was recorded. If

the project later adds parallel review or a loop, it can name the same boundaries instead of relying on an invisible convention in an agent prompt.

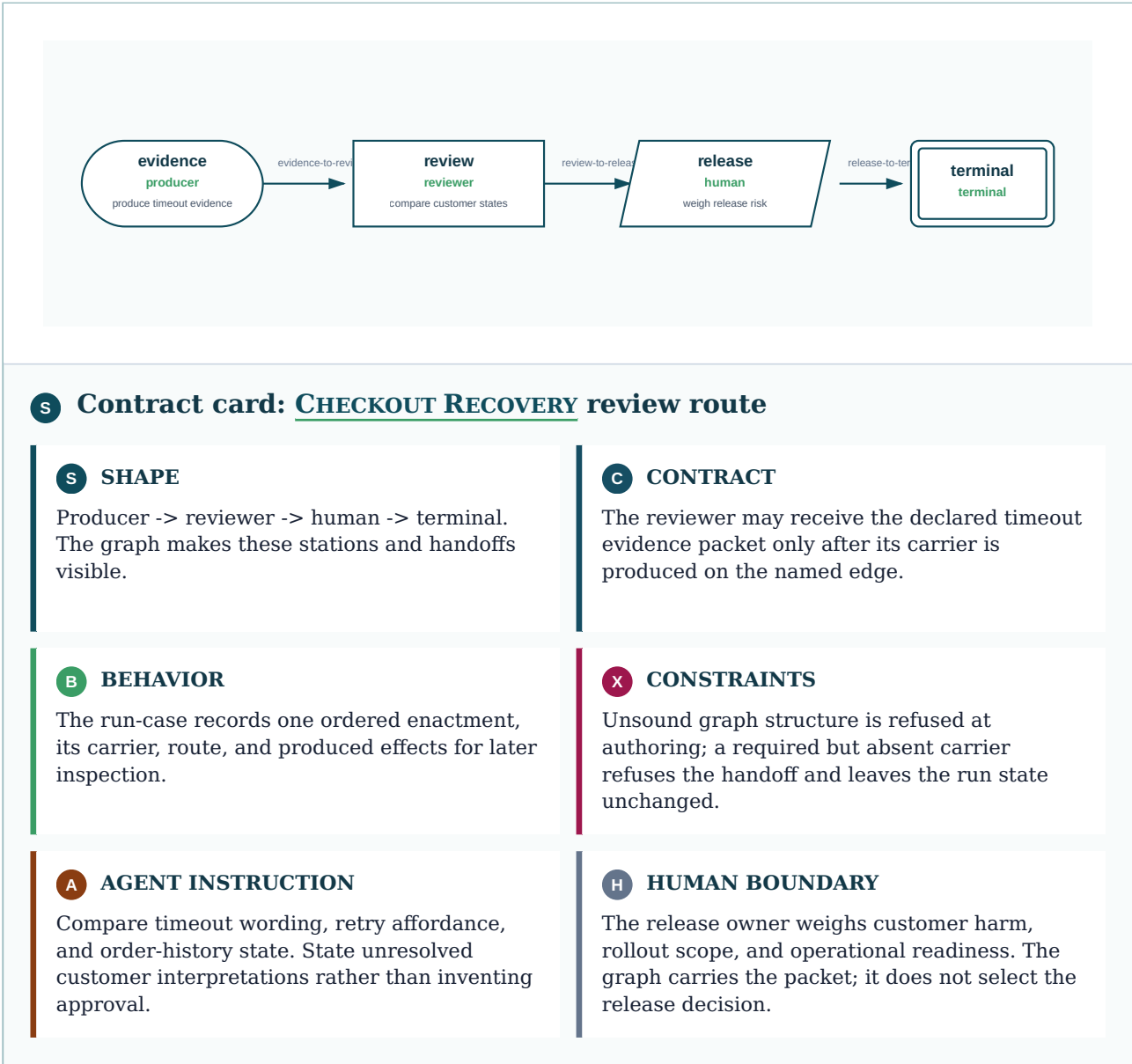


Figure 7. **CHECKOUT RECOVERY** review route. The checked-in JSON fixture was accepted and linted clean by the current Lettuce binary; *lettuce graph-def viz* generated the adjacent Mermaid source. This PDF SVG is a static rendering of that generated topology, using Lettuce concern shapes rather than a separate diagram language. The declared *timeout-evidence* carrier is present in the fixture but not rendered by the current visualization, so the picture is not the whole edge contract.

Behavior before shape. Do not begin a graph by arranging node icons. Begin by naming the claimed handoff, the worker instruction, the evidence that crosses the edge, and the condition that may refuse progress.

What the system can refuse

The current graph authoring contract checks structural properties before storing a definition. A missing start node, dangling edge, unreachable node, or uncapped cycle is refused rather than accepted as a diagram that might fail later. Forks, joins, quorums, bounded loops, router guards, and execution-policy fields have their own declared validity rules. That gives the graph a stronger status than a whiteboard sketch: some impossible coordination shapes are rejected at the point they are authored.

At runtime, a run-case can also refuse a declared handoff. A carrier is a write-once, content-addressed datum emitted on an edge visit. When an advance declares that a carrier must already have been produced, the system reads that fact back; without it, the advance is refused and the run state remains unchanged. A router chooses among its declared edges from recorded run-case facts, not from an executor simply claiming that a preferred branch won.

For [CHECKOUT RECOVERY](#), this makes a useful small promise: the review station cannot be recorded as having received the required evidence packet if the packet was never produced on the declared edge. The promise is not that the packet was adequate, that the reviewer read it carefully, or that the owner made the right release choice.

A human node is not a human decision

This distinction is the point at which many agent systems become misleading. A graph can route work toward a human-shaped station. It can preserve the input carrier, record the transition, and make a missing prerequisite loud. It cannot infer the person's authority, enumerate the alternatives they considered, or judge whether their rationale was sound.

In the [CHECKOUT RECOVERY](#) case, the graph should therefore deliver a concise decision packet, not manufacture an approval. The packet contains the timeout evidence, the reviewer finding, the known alternatives, and the claim that remains unsupported. The release owner decides whether to hold, constrain, or release. The decision chapter will make that act and its later supersession explicit; the graph chapter only establishes how the relevant work became inspectable.

Failure mode: topology theater. A detailed graph can look more governed than it is. If its behavioral instructions, evidence handoffs, authority boundaries, and refusal conditions are implicit, it is a picture of confidence, not an executable coordination contract.

‡ What the reader should keep

1. **A graph definition is a contract, not a flowchart.** Its stored, versioned shape can refuse selected invalid coordination structures.
2. **Concern, instruction, and execution policy are distinct.** Know what a station is, what its worker should do, and how the orchestrator intends to staff it; current run-cases do not enforce that staffing intent.
3. **Carriers make a declared handoff inspectable.** They do not establish that the handed-off material is adequate for a larger claim.
4. **A run-case records one enactment.** It can prove selected transition and refusal facts, not that the process deserved to end successfully.
5. **A human node is not an authority model.** Route evidence to people; do not turn a diagram label into a false decision receipt.

REFLECTION

If every node label disappeared, which edge contract or refusal would still make this graph worth storing?

The Shape Catalog

Claim status: shipping `graph-def` authoring and visualization vocabulary verified against Lettuce `main@165de048e` on 2026-08-10. Shape determines a structural concern and selected validation/runtime rules. It does not supply an instruction, authority model, or semantic result.

The question

When a reader sees a stadium, rectangle, diamond, hexagon, parallelogram, trapezoid, nested box, or cylinder in a Lettuce graph, what may they safely infer before reading the labels and contract?

The thesis

Shape is a compact type annotation. It lets a reader scan the coordination role and lets the product attach narrow structural rules. The same shape can host very different behavior. A catalog is useful only when it teaches both the inference the shape permits and the inference it forbids.



The ten graph shapes rendered by Lettuce: producer, reviewer, verifier, router, gate, human, fork, join, terminal, and subgraph.

Figure 8. The Lettuce shape vocabulary. Shipping visualization maps nine node concerns and the reusable *uses* seam to these forms. Reviewer and verifier share rectangular geometry but remain distinct concerns. Text and line style remain necessary; color is not the only carrier of meaning.

Work stations

Producer



Reviewer



S Reviewer / rectangle

S SHAPE

A rectangle with reviewer concern marks a station that examines work and may surface findings or dissent.

C CONTRACT

The station occupies a review lane in the route; its arrival can contribute to a later join.

B BEHAVIOR

The instruction defines the review lens: architecture, UX, security, operations, adversarial attack, or another question.

X CONSTRAINTS

Completion does not mean approval. Several reviewers do not become independent merely because they occupy distinct nodes.

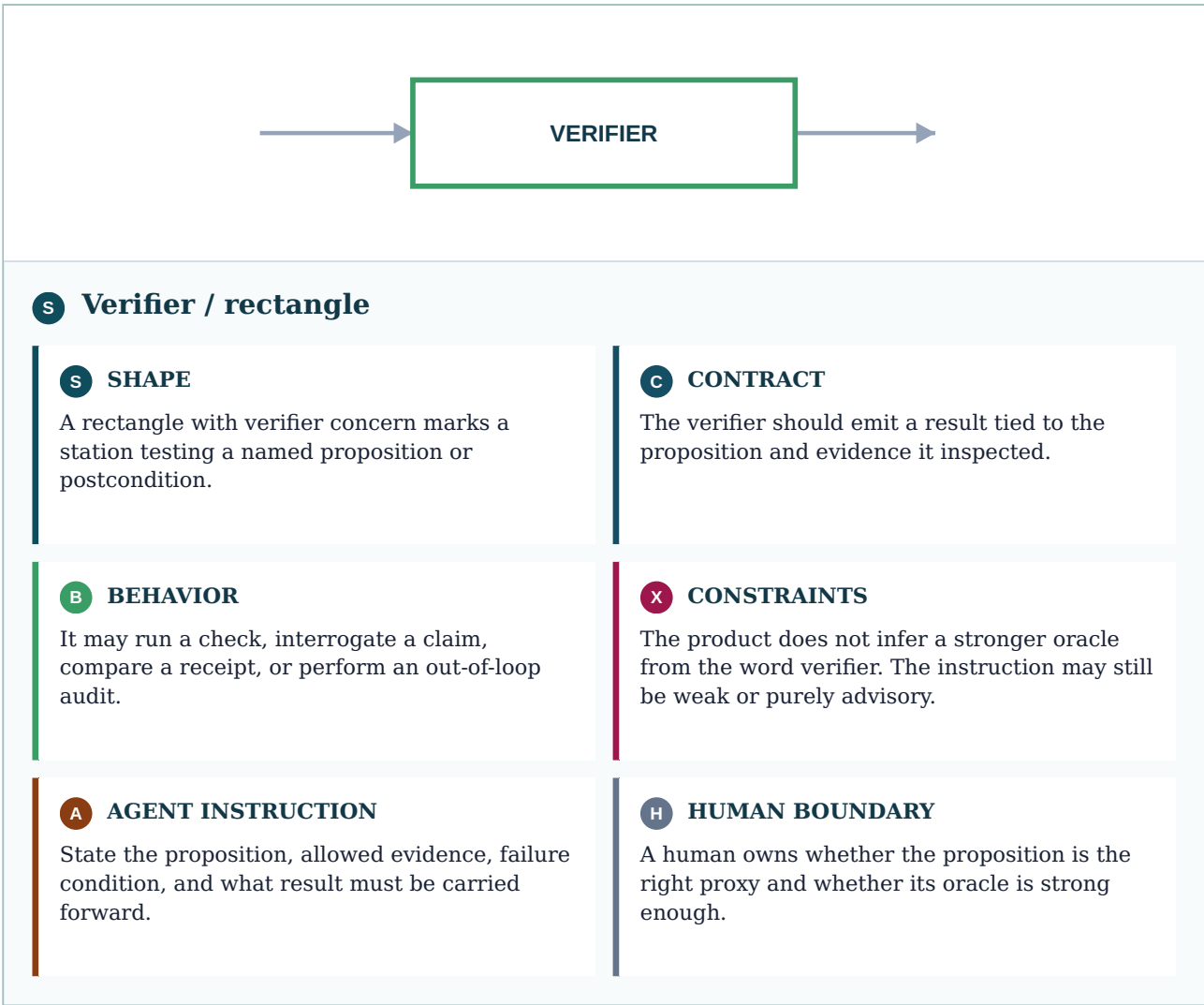
A AGENT INSTRUCTION

Give the reviewer a falsification question, evidence access, expected finding format, and explicit permission to dissent.

H HUMAN BOUNDARY

A person chooses whether the review lens and reviewer diversity match the consequence.

Verifier



Reviewer and verifier share a rectangle because both are behavioral work stations. Their concern labels remain load-bearing. A reviewer explores and challenges under a lens; a verifier tests a declared proposition. A team should not choose between them for visual variety.

Decision and admission stations

Router



S Router / hexagon

S SHAPE

A hexagon marks conditional selection among outgoing edges.

C CONTRACT

Exactly one route must match guards over recorded carrier, wave-outcome, or effect facts; one unguarded default may exist.

B BEHAVIOR

The runtime evaluates the supplied guard set against the run ledger and records the chosen route and consumed facts.

X CONSTRAINTS

Ambiguous, unsatisfied, or unmatched routing is refused. The router cannot inspect arbitrary world state or invent a missing fact.

A AGENT INSTRUCTION

Produce the decision fact before routing. Keep the default route safe and make escalation require positive evidence.

H HUMAN BOUNDARY

A person decides which facts justify each route and whether default exit or escalation is safer.

Gate



S Gate / diamond

S SHAPE

A diamond marks an admission boundary before work may proceed.

C CONTRACT

The enacted prerequisite must succeed; a failed task requirement, missing carrier, or other declared rule leaves protected state unchanged.

B BEHAVIOR

The instruction explains what should be evaluated; runtime teeth come from the concrete gate and advance parameters.

X CONSTRAINTS

A diamond is not itself a predicate. Without a biting rule it is visual intent, not enforcement.

A AGENT INSTRUCTION

Name the prerequisite, its evidence, the refusal code or state, and the recovery path.

H HUMAN BOUNDARY

A human decides whether the predicate is sufficient and whether an exception may exist.

Human



S Human / parallelogram

S SHAPE

A parallelogram marks a station intended for human work or judgment.

C CONTRACT

The route makes the human boundary visible and can preserve arriving and leaving provenance.

B BEHAVIOR

The instruction may ask for review, approval, risk acceptance, advice, or another act; these behaviors are not interchangeable.

X CONSTRAINTS

The shape does not define identity, authority, evidence requirements, choices, timeout, escalation, or a durable decision receipt.

A AGENT INSTRUCTION

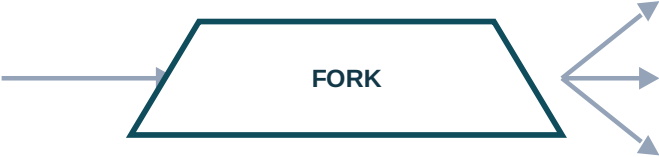
Prepare the question, alternatives, evidence, dissent, consequence, deadline, and safe unresolved route.

H HUMAN BOUNDARY

The named accountable person performs the semantic act; the system must not convert silence or timeout into approval.

Structural stations

Fork



S Fork / widening trapezoid

S SHAPE

A widening trapezoid marks one flow opening into at least two branches.

C CONTRACT

Each outgoing edge can open a distinct content-addressed branch while main state remains at the fork.

B BEHAVIOR

Branches advance against their own tips and preserve separate arrival identity.

X CONSTRAINTS

A degenerate fork is refused. Separate branches do not imply concurrent clocks or independent reasoning.

A AGENT INSTRUCTION

Give every branch a distinct purpose, evidence boundary, and expected carrier.

H HUMAN BOUNDARY

An operator chooses useful decomposition and controls correlated work or resource pressure.

Join



S Join / narrowing trapezoid

S SHAPE

A narrowing trapezoid marks branch reconvergence.

C CONTRACT

The join fires exactly once after K distinct inputs arrive; omitted or zero quorum means all M inputs.

B BEHAVIOR

Arrival coordinates are folded into run state and concurrent identical fires de-duplicate.

X CONSTRAINTS

Below-quorum firing is refused. Arrival counts completion under the run contract, not agreement or correctness.

A AGENT INSTRUCTION

Define what arrival means, preserve dissent, and justify K against consequence rather than convenience.

H HUMAN BOUNDARY

A person owns the threshold and whether minority evidence requires escalation despite quorum.

Terminal



S Terminal / nested rectangle

S SHAPE

A nested rectangle marks the sink of a bounded route.

C CONTRACT

A run-case may close from the terminal and preserve its outcome in the event ledger.

B BEHAVIOR

It absorbs control after the graph's declared work is complete or has reached a bounded failure exit.

X CONSTRAINTS

A terminal with outgoing edges is a lint smell. Reaching it does not complete a task, satisfy DoD, or approve external release.

A AGENT INSTRUCTION

Record a narrow outcome and link effects; do not translate terminal arrival into a broader success claim.

H HUMAN BOUNDARY

A human interprets whether the bounded process result is sufficient for the next product decision.

Subgraph



S Subgraph / cylinder

S SHAPE

A cylinder marks a **uses** node that references a versioned graph and binds its open ports.

C CONTRACT

The reference is pinned; compile expands the transitive graph and hashes the effective result.

B BEHAVIOR

The visible node collapses a reusable graph; a seam execution-policy default may fill blank worker policies inside it.

X CONSTRAINTS

Missing definitions, cycles, unbound ports, and unsound expansion are refused. Default visualization does not expose the interior.

A AGENT INSTRUCTION

Inspect the pinned child, compiled hash, port bindings, and seam defaults before trusting the collapsed name.

H HUMAN BOUNDARY

A project owner decides whether the reused pattern, version, cost, and authority model fit this context.

How to read any unfamiliar node

Use the same sequence every time:

1. Read the geometry as a structural concern.
2. Read the concern label to disambiguate shared shapes.
3. Read the instruction for behavior.
4. Read execution policy for staffing and context lifetime.
5. Read edges for carriers, guards, loops, and handoffs.
6. Read the contract card for constraints and human boundary.

⚠ FAILURE MODE: SHAPE COMPLETION

A reader sees a diamond and supplies an imagined gate, sees a human parallelogram and supplies imagined approval, or sees three rectangles and supplies imagined independent review. The diagram becomes more complete in the reader's head than it is in the stored contract.

≠ What the reader should keep

1. Shape is a type hint with narrow structural consequences.
2. Concern, instruction, execution policy, and edge contract remain separate.
3. Shared geometry does not erase concern differences.
4. Human and gate shapes are boundaries, not complete authority or predicates.
5. A collapsed subgraph must be inspected by version and compiled identity.

🕒 REFLECTION

Which meaning did you infer from the last graph shape you saw before reading its concern, instruction, edge contract, and authority boundary?

Run-Cases, Carriers, and the Ledger of an Attempt

Claim status: shipping product behavior verified against Lettuce `main@165de048e` on 2026-08-10. Graph definitions and run-cases are currently local or dedicated-Git only; the HTTP surface is deferred.

? The question

A graph definition says which coordination shape is allowed. What lets a later reader determine which path one attempt actually took, what crossed each handoff, and which project objects it changed?

💡 The thesis

A run-case is not a log attached to a diagram. It is an event-sourced enactment: its state, revision, visited trail, branches, route facts, loop outcomes, and telemetry are derived from recorded events. A carrier is a write-once datum at a particular handoff. An effect is a declared link from an advance event to an object the worker changed. Together they turn execution into a traversable ledger.

They do not prove the worker told the truth. The caller supplies carrier values, effect declarations, duration, and cost. Lettuce preserves and cross-links those claims; it does not secretly observe the world to certify them.

Definition, attempt, handoff, consequence

Four objects answer four different questions:

Object	Question	Durable fact
Graph definition	What route was declared?	Versioned shape, behavior, execution policy, and effective hash.
Run-case	What happened in this attempt?	Event-derived state, revision, visited path, and recorded outcomes.
Carrier	What crossed this edge on this visit?	Content-addressed value plus typed references.

Object	Question	Durable fact
Effect	What object did the step say it changed?	Kind, object reference, and optional carrier back-reference on the advance event.

The separation matters. Putting a screenshot reference in a carrier does not make the screenshot the graph. Recording **cell-hardened** as an effect does not perform or validate the cell transition. Closing a run-case does not close its backing task or approve a release. The ledger is useful because these objects can be related without being collapsed.

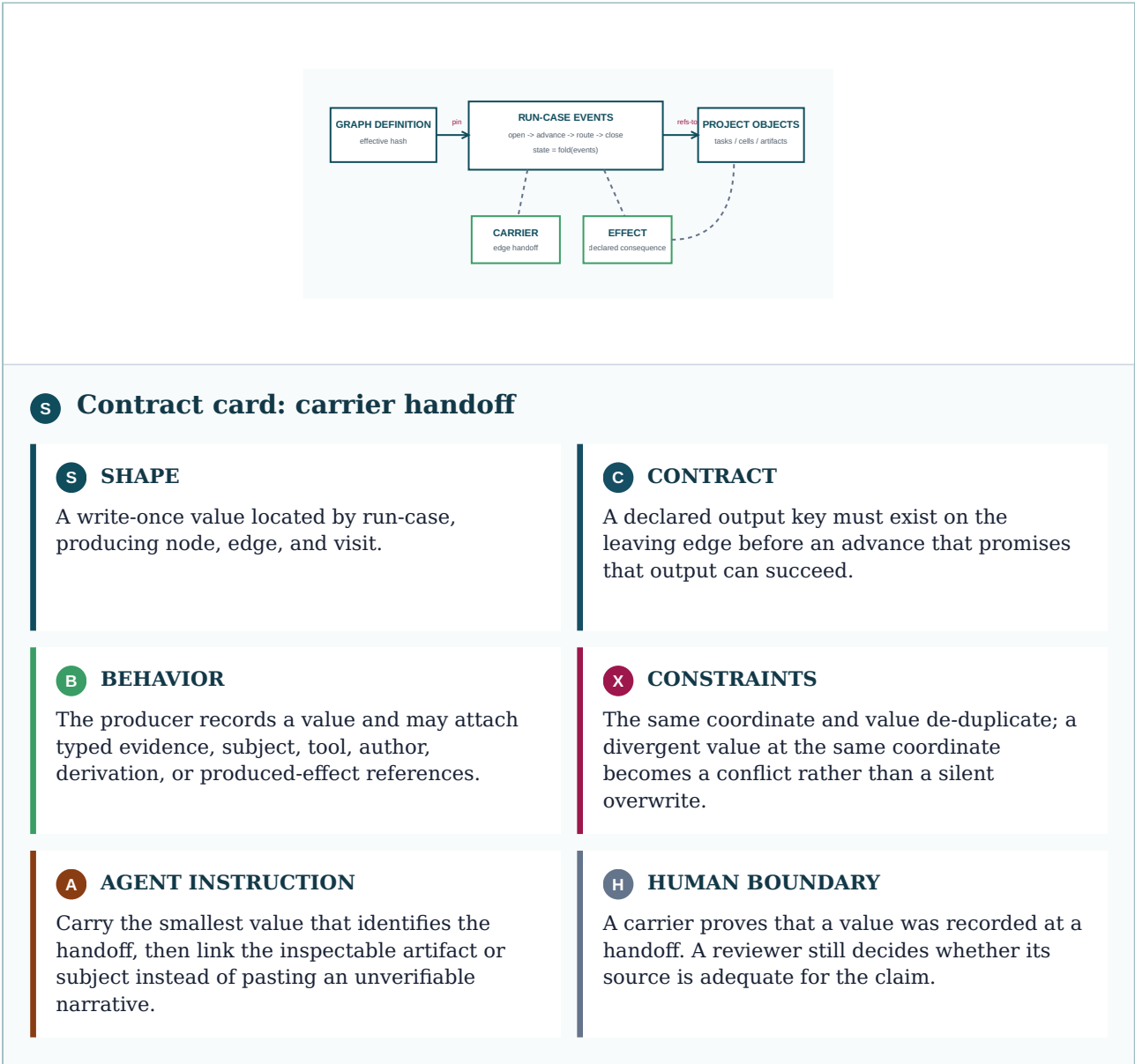


Figure 9. The ledger of one attempt. Shipping behavior: a new run opened against its required stored graph pins the effective hash; run state folds from events, carriers are content-addressed handoffs, and effect references support reverse lookup. Dashed lines are provenance links, not control flow.

CHECKOUT RECOVERY: the missing packet bites

The **CHECKOUT RECOVERY** graph expects an evidence packet before timeout review. The worker opens a run against the stored graph and reaches the producer. If it tries to advance while declaring that it produced the timeout packet, Lettuce reads the carrier ledger first. With no carrier at that edge and visit, the advance is refused and the run state remains unchanged.

That refusal is more valuable than a warning. It prevents the next station from being represented as reached under the declared handoff contract. The producer then creates a carrier whose value points to the screenshot set and attaches typed references to the backing task and artifact. Retrying the same advance can now succeed. The reviewer receives an inspectable address, not proof that the screenshots cover every customer state.

The transition may also record a produced effect such as an artifact addition or task transition. The reverse-reference view later inverts that edge: given the object, it lists the run-cases and transitions that declared they touched it. A cell’s reference view surfaces the same reverse provenance. This is how the work graph and quality map become navigable in both directions.

Recorded, delivered, and received are different states

During Lettuce’s own development, an agent confirmed a real finding and recorded it in its local tracking row. It then sent two batch handoffs. The finding appeared in neither. From inside the producer’s loop, “I recorded it” felt like “I delivered it” because the local row was complete. From the recipient’s side, the finding did not exist.

That failure is a useful model for every carrier and effect ledger. A producer record can establish that a value was created and made available at a named handoff. It cannot, by itself, establish that the intended consumer fetched it, understood it, incorporated it, or caused the outside-world consequence. Those are later states with different possible receipts.

State	Strongest narrow claim	Missing proof
Recorded	The producer durably wrote the handoff value.	That the value left the producer’s context.
Delivered	A consumer-controlled boundary accepted the value.	That the consumer understood or acted on it correctly.
Acknowledged	The intended consumer identified what it received.	That the downstream semantic or external outcome was good.

The mitigation is not to call every carrier delivered. For consequential handoffs, require a receipt issued on the receiving side, reconcile the set of items promised against the set

actually received, and leave absence as unknown or undelivered. A sender cannot close that distinction by signing both sides.

Reader's test. Whenever a ledger says something was handed off, ask which side issued the receipt. Producer evidence proves production; consumer evidence is needed for receipt.

Replay means refolding, not reenacting the world

`graph-run-case show` derives the current projection from committed events. `graph-run-case viz` overlays current and visited nodes on the graph. Route decisions preserve the facts consumed; wave closures preserve clean/progress and queue outcomes; recorded duration and exact micro-USD totals fold from events. A later reader can reproduce the ledger's conclusion.

That is **state replay**. It is not a fresh rerun of tests, a second model call, or a recreation of an external deployment. If an event says a worker observed 1,250 milliseconds, replay preserves that observation. It does not make the clock run again. Reproducibility therefore has layers:

1. the same events derive the same run projection;
2. the pinned graph hash resolves to the exact stored specification version;
3. referenced artifacts make inputs inspectable;
4. external effects may still require independent records or reenactment.

The board follows the same provenance discipline: its source revision and `generated_at` value derive from the latest store event, not the reader's render clock. With unchanged store state, repeated exports can remain byte-identical. That supports reproducible inspection; it does not make an old projection fresh with respect to an external system.

The binding seam

The current runtime deliberately remains graph-definition-free **after open**. A new open must resolve a stored definition and its declared start, and records the effective-hash pin. Historical pre-pin runs remain readable because absence is history, not corruption. At advance time, however, the caller supplies effective produced-output keys, quorum, loop parameters, and router guard sets. The engine enforces the supplied contract and records its outcomes, but it does not derive the entire advance contract from the pinned definition.

FAILURE MODE: DECLARED AND ENACTED GRAPHS DIVERGE

A pinned run can record weaker runtime parameters than the definition declares. Pinned-version conformance can expose selected divergence afterward, but operators must still treat the orchestrator and any pre-effect conformance gate as part of the trusted control surface.

This does not make the runtime refusals fictitious. A supplied quorum still bites; a supplied carrier requirement still refuses; an ambiguous supplied route still fails. `graph-run-case conform` resolves the pin to the exact graph version and replays the recorded walk, reporting undeclared edges, unknown nodes, below-declaration quorum, and a finished walk that reached terminal without ever firing its declared join. It is a read-time report with a non-zero exit on non-conformance, not a mutation refusal.

Terminal state also gained a necessary distinction. A run may be completed, failed, or abandoned. Abandonment requires a reason, remains visible, and is excluded from conformance rather than being counted as a process that ran. Closing does not delete a failed attempt or turn it into positive evidence.

What the reader should keep

1. A definition declares a coordination contract; a run-case records one attempt.
2. Carriers preserve handoffs; effects preserve declared consequences.
3. A producer-side carrier does not prove consumer-side delivery or understanding.
4. A refused handoff is evidence that a narrow prerequisite actually bit.
5. Replay reconstructs recorded state, not the external world. A pin plus conformance makes route divergence inspectable; preventive authority still requires conformance to gate the consequential effect.

REFLECTION

For the most consequential effect in one run, which receipt comes from the producer, which comes from the receiving system, and which does not yet exist?

Parallelism, Quorum, and Bounded Convergence

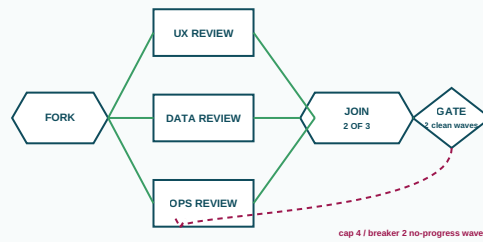
Claim status: shipping graph-definition and CLI run-case behavior, interpreted through a review case. The quality of reviewers and their independence remain outside the mechanical contract.

The question

If several agents review the same change, when may their branches reconverge, and how can an iterative review stop without pretending that repetition itself creates truth?

The thesis

Parallelism is a coordination shape, quorum is a counting rule, and convergence is a declared stopping oracle. Lettuce can enforce their structural and event contracts: branches must arrive, thresholds must be met, loops must be capped, and exits must agree with recorded wave outcomes. None of those rules establishes that the reviewers were independent, competent, or aimed at the right question.



S Contract card: fork, quorum join, and loop

S SHAPE

A fork opens distinct branch coordinates; a join merges arrivals; a capped back-edge creates repeated waves.

C CONTRACT

The join fires once at K distinct arrivals. The loop exits only after its recorded floor, stability, drain, or breaker condition permits it.

B BEHAVIOR

Branch tips, arrived sets, wave outcomes, and the final exit are folded from immutable run events.

X CONSTRAINTS

Unsatisfied joins, premature exits, cap overruns, and continued work after a tripped breaker are refused.

A AGENT INSTRUCTION

Give branches genuinely different questions or evidence access. Define “clean,” “progress,” and “level-up” before the first wave.

H HUMAN BOUNDARY

A person owns the adequacy of K, the independence of reviewers, and whether the stopping oracle matches the risk.

Figure 10. Bounded review convergence. Shipping shape and runtime primitives: content-addressed branches, K-of-M join, wave outcomes, stability floor, hard cap, and circuit breaker. The labels describe coordination facts, not semantic quality.

Parallel branches are not repeated prompts

At a fork, opening a branch leaves the main state at the fork while creating a content-addressed branch for one outgoing edge. Other edges can open their own branches. Each branch advances against its own tip. The join can fire only after the required number of distinct branches has arrived.

For CHECKOUT RECOVERY, three branches might ask different questions:

- Can the timeout message distinguish pending, failed, and completed payment?
- Does order history converge to the same state after retry and refresh?

- Can an operator detect and repair a duplicate-payment incident?

This is stronger than asking three workers “is checkout good?” The questions create different opportunities to falsify the release claim. Yet even that design does not guarantee independence. Three agents using the same model, prompt framing, code snapshot, and test oracle may share one blind spot.

What a quorum establishes

A two-of-three join establishes that at least two distinct branch coordinates arrived under the enacted run contract. It may also preserve which coordinates were present when the join fired. It does not establish:

- that two independent arguments support the same conclusion;
- that the dissenting branch was safely irrelevant;
- that reviewers saw different evidence;
- that the threshold was appropriate to the consequence; or
- that “arrival” meant approval rather than completion.

⚠ FAILURE MODE: CONSENSUS LAUNDERING

A quorum converts branch arrival into a mechanical admission rule. Calling that result “truth,” “quality,” or “independent review” silently adds premises the join never checked.

Quorum is useful when its proposition is stated narrowly: “two required review lanes completed and produced their declared handoffs.” The release owner can then inspect the results, including dissent, instead of receiving a flattened vote count.

A loop needs three different limits

An iterative review can fail by stopping too early, running forever, or repeating without learning. Lettuce separates these concerns:

Primitive	Question it answers
<code>min_times</code>	Have we performed the minimum number of waves?
<code>until_stable</code>	Have N consecutive waves closed cleanly without a level-up finding?
<code>until_drain</code>	Does the latest wave say the declared work queue is empty?
<code>cap</code>	

Primitive	Question it answers
	What is the absolute maximum number of back-edge visits?
<code>circuit_breaker</code>	Have M consecutive waves made no declared progress?

A wave closure records an outcome. A “clean” wave extends the stability run; a level-up finding resets it because the problem model changed; progress resets the non-progress streak; a remaining queue of zero can satisfy a drain oracle. The loop exit folds these events and refuses a premature claim.

This is honest only if “clean,” “progress,” and “queue” have operational meanings. If an agent can mark any wave clean to escape, the event is still durable but the oracle is weak. A breaker is not a successful convergence signal either. It permits a bounded failure exit so stalled work can escalate instead of consuming resources indefinitely.

Choosing the bar from consequence

The number of agents is a poor starting point. Begin with the failure:

1. What mistake must this review be able to discover?
2. Which evidence or perspective would expose it?
3. Which failures are correlated across workers?
4. What consequence follows from a false positive?
5. Which condition warrants escalation instead of another wave?

For a reversible wording experiment, one focused reviewer and a deterministic check may be enough. For payment recovery, the branches should separate customer comprehension, transaction consistency, and operations. For an irreversible migration, a distinct human authority may be mandatory even after unanimous agent review.

✚ What the reader should keep

1. Forks create branch identity; they do not create intellectual diversity.
2. A quorum counts arrivals under a contract; it does not count truth.
3. Convergence requires a defined outcome, not merely repeated execution.
4. Caps and breakers bound cost and failure; they are not success signals.
5. Review topology should follow the consequence and likely correlation of error.

REFLECTION

What shared blind spot could make every branch agree, and which distinct evidence or falsification lens would expose it?

Composition Without Opacity

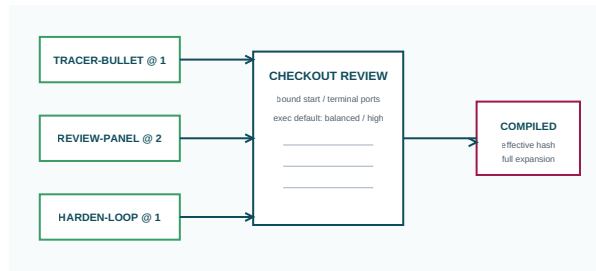
Claim status: shipping graph-definition, catalog, versioning, compilation, lint, and visualization behavior. Expanded visualization of nested subgraphs remains deferred.

The question

How can a team reuse a proven review pattern without turning its workflow into a name whose internal behavior, version, and cost assumptions are invisible?

The thesis

Reuse is safe only when the seam is explicit. Lettuce composes graph definitions through version-pinned **uses** references, bound ports, deterministic expansion, and an effective hash over the whole executable result. Catalog patterns provide named, verified starting shapes. The name aids discovery; the pin and expansion carry reproducibility.



S Contract card: version-pinned subgraph

S SHAPE

A composite node references `slug@version` and binds the reused graph's start and terminal ports into its parent.

C CONTRACT

Compilation must resolve an acyclic, sound expansion. The stored version and effective hash identify the result.

B BEHAVIOR

The referenced graph is inlined with namespaced nodes; explicit child policies survive and seam defaults fill only missing knobs.

X CONSTRAINTS

Missing definitions, cycles, unbound ports, unsound expansions, and invalid placement of execution defaults are refused.

A AGENT INSTRUCTION

Inspect the catalog pattern and compiled hash. Pin the intended version; do not infer behavior from the pattern name alone.

H HUMAN BOUNDARY

An owner decides whether the reused behavior, cost ceiling, and review authority fit this project's consequence.

Figure 11. A visible composition seam. Shipping behavior: stored uses references are pinned to concrete versions, compilation expands the transitive closure, node identifiers are namespaced, and the effective hash covers the resulting topology, instructions, guards, and execution policies.

Names are for discovery; hashes are for identity

The shipped catalog contains patterns such as a gate, review panel, claim-verification route, audit wave, hardening loop, red team, and composed deep review. Each catalog entry has a version, description, worst-case ordinal cost hint, sound specification, and effective hash. Materializing a pattern creates an ordinary project graph definition whose compiled hash matches the catalog entry.

That gives a useful three-step discipline:

1. discover by a human name;
2. inspect the versioned contract;
3. reproduce by the compiled identity.

“Use deep review” is not enough. Which deep review version? Which atomic patterns does it pin? What instructions staff its reviewers? Which quorum and loop policies apply? What is the worst-case cost band? The catalog can answer those questions before enactment.

Pinning prevents ambient upgrades

When a definition is authored or revised, a floating child reference is rewritten to the concrete latest version it resolved to at that time. Earlier versions remain append-only. A parent pinned to `review-panel@1` therefore does not silently acquire `review-panel@2` when the child evolves.

This is the right default for auditability. Reuse should make improvement easier without making history ambiguous. Adopting the newer child is an explicit parent revision, producing a new expansion and effective hash. A later reader can see that the process changed.

Policy defaults belong at the seam

A reusable atomic graph knows its behavior but may not know how expensive its workers should be in every product tier. A composite may therefore supply an execution-policy default for the behavioral stations in its subtree. The default fills missing model, effort, or residency fields; it does not overwrite an explicit child choice. Nested defaults use the nearest applicable seam.

This is a subtle but important form of composability. The atom remains reusable; the parent can staff it for its context; and the compiled result, not an invisible runtime override, receives the effective identity.

That identity is presently stronger than the runtime guarantee. The policy is authored, linted, displayed, and hashed, but Lettuce’s run-case engine does not consume it to choose or constrain a worker. An external orchestrator must honor the declared staffing contract, and an audit must distinguish its receipt from the fact that the declaration existed.

Lint protects judgment without pretending to be soundness

Authoring refuses structurally invalid graphs. Lint reports legal but suspicious ones: a split without a join, a terminal with an outgoing edge, a dead-end gate, a review edge without an evidence carrier, a behavioral node without an instruction, or a looping worker whose residency is implicit.

⚡ REFUSAL AND SMELL ARE DIFFERENT CONTRACTS

Soundness says the graph cannot be stored or compiled as written. Lint says the graph is executable but deserves review. Keeping those channels distinct prevents advice from masquerading as law.

The distinction echoes the whole book. Some properties have deterministic oracles and should bite. Others are patterns of risk and should remain visible to an accountable reviewer. Turning every smell into a fatal rule would make the catalog rigid; turning every rule into advice would make the graph decorative.

The present visualization limit

`graph-def viz` deterministically renders the stored graph in Mermaid or DOT. It shows structural concern, shortened behavior, execution policy, quorum, loop controls, and router guards. A subgraph appears as one cylinder with its versioned reference and seam default. An expanded visualization that exposes the entire inlined subtree is deferred.

That means the default picture is an abstraction boundary, not a full proof of the expansion. For audit, pair it with `graph-def compile` and `graph-def show`. The picture supports comprehension; the compiled effective form supports identity.

⚠ FAILURE MODE: PATTERN-NAME AUTHORITY

A familiar catalog name can acquire more trust than its actual instruction, version, or applicability deserves. Reuse reduces authoring work; it does not transfer accountability from the project owner to the catalog.

≠ What the reader should keep

1. Discover patterns by name, inspect by contract, and reproduce by hash.
2. Version pins make upgrades explicit and history stable.
3. Seam defaults fill policy blanks; they must not erase child intent.
4. Soundness refusals and design-smell warnings serve different purposes.
5. A collapsed subgraph picture is a useful boundary, not the full expansion.

REFLECTION

Which project-specific authority or failure assumption would remain hidden if you adopted this pattern by name without inspecting its pin and expansion?

The Graph Pattern Catalog

Claim status: shipping embedded catalog verified against Lettuce `main@165de048e` on 2026-08-10. The binary exposes 16 immutable name/version entries across 13 pattern families. Catalog cost hints are authoring metadata describing worst-case ordinal cost; runtime control does not read them.

The question

When should a team reuse a shipped graph pattern, which version should it pin, and what does the pattern still require the project to decide?

The thesis

A graph catalog is a library of coordination contracts, not a menu of quality certificates. It reduces the cost of authoring sound topology and makes proven composition idioms inspectable. The project still owns applicability, instructions in context, evidence requirements, reviewer diversity, human authority, and enactment.

Three different objects

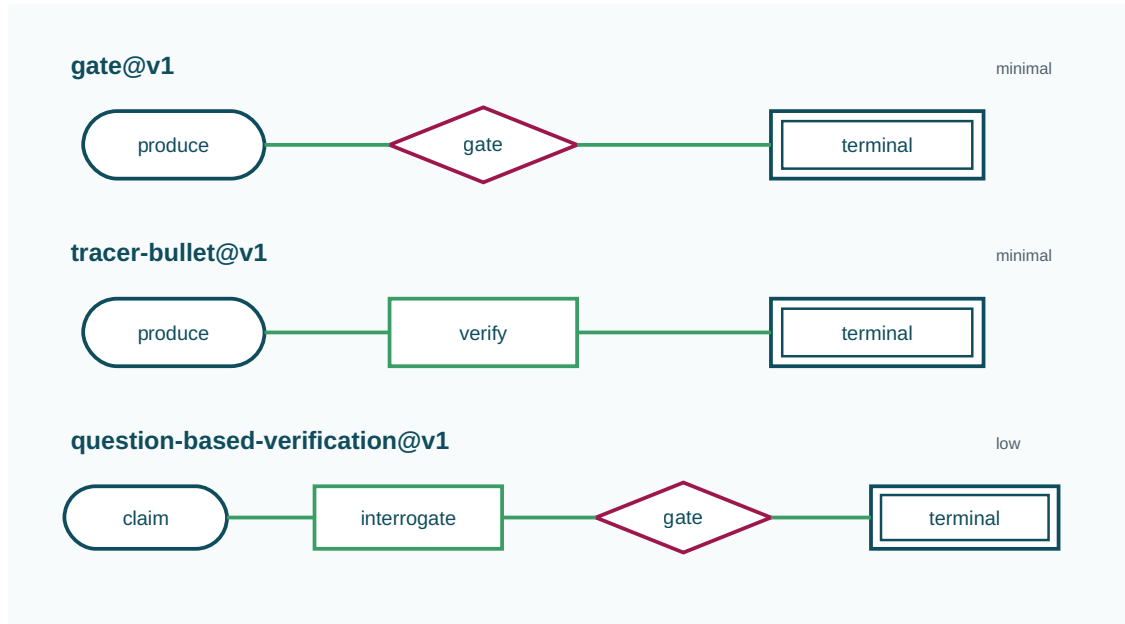
The word catalog can blur three layers:

Layer	Where it lives	What it means
Embedded pattern	In the binary	An immutable, named, versioned, sound graph specification with description, worst-case cost hint, and effective hash.
Materialized graph definition	In a project store	A project-local graph created from a pattern, revisable and composable like any other stored definition.
Run-case	In a project store	One event-sourced enactment associated with a graph definition or graph label.

Catalog reads inspect the embedded library without an initialized store. Materializing a named version creates one project-local definition. The same soundness and hash path used by ordinary graph authoring applies; a faithful materialization compiles to the catalog hash.

Do not say a project “uses review-panel” merely because its binary ships the pattern. Use begins when the project materializes, specializes, enacts, or otherwise explicitly adopts that contract.

Family 1: minimal atoms



The gate, tracer-bullet, and question-based-verification patterns.

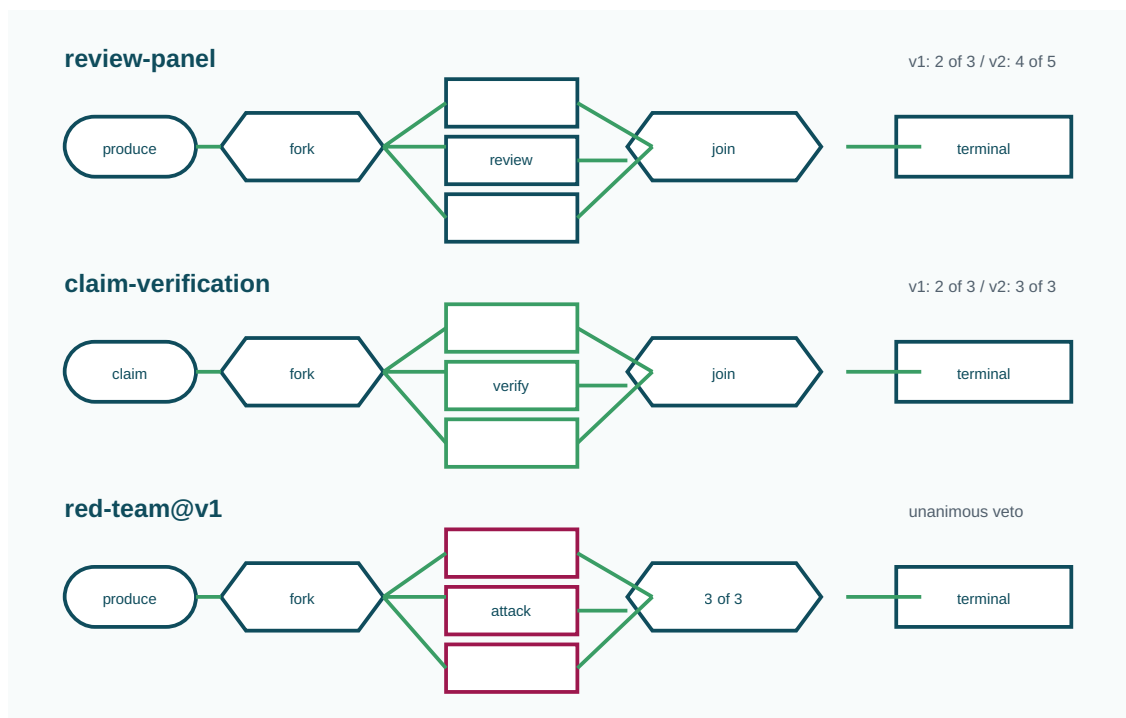
Figure 12. Minimal catalog atoms. These are small, reusable coordination contracts. The producer, verifier, gate, and terminal labels summarize concern; the stored catalog entry also carries full instructions.

Pattern	Cost	Mechanical shape	Choose it when	It does not establish
gate@v1	minimal	producer -> gate -> terminal	One produced handoff must cross one explicit admission boundary.	That the gate predicate is sufficient or the produced work is good.
tracer-bullet@v1	minimal	producer -> verifier -> terminal	You need a thin end-to-end claim and one verification lane before building a larger process.	Broad coverage, adversarial review, or product readiness.
question-based-verification@v1	low	producer -> interrogator/ verifier -> gate -> terminal	A claim should survive explicit probing questions before admission.	That the questions are complete, independent, or answered truthfully.

The tracer bullet is the right starting pattern surprisingly often. It tests whether the project can carry one real input through one real check into one inspectable result. A large review graph built before that path works can hide basic integration failures under orchestration complexity.

Question-based verification is useful when a binary check is too narrow but the claim can be interrogated. Its verifier instruction is the behavior. The gate does not manufacture a “must answer” oracle from the topology alone; the project must define what constitutes a satisfactory answer and which evidence is required.

Family 2: panels and adversarial lanes



Review-panel, claim-verification, and red-team pattern families.

Figure 13. Panel families. All share producer → fork → parallel work → join → terminal. Their worker concern, instruction, quorum, and purpose differ. Version changes preserve older contracts rather than silently raising the bar.

Review panel

review-panel@v1 opens three reviewer branches and joins at two of three. **review-panel@v2** opens five and joins at four of five. The newer version raises both reviewer count and threshold; it does not mutate projects pinned to v1.

Use a review panel when distinct lenses should examine one produced object and a declared subset may admit reconvergence. Specify those lenses. Five copies of the same generic approval prompt are a more expensive single instrument.

Claim verification

`claim-verification@v1` forks one claim to three verifier stations and joins at two of three. `claim-verification@v2` requires all three. Unlike a reviewer, each verifier should test a named claim rather than offer broad design opinion.

The v2 unanimous bar is mechanically stronger but semantically ambiguous unless dissent is preserved. “All branches arrived” can mean all checks completed; it does not necessarily mean all verifiers endorsed the claim.

Red team

`red-team@v1` forks work to three adversarial reviewer stations and requires three-of-three arrival. The catalog describes this as veto semantics: every attack branch must clear before reconvergence.

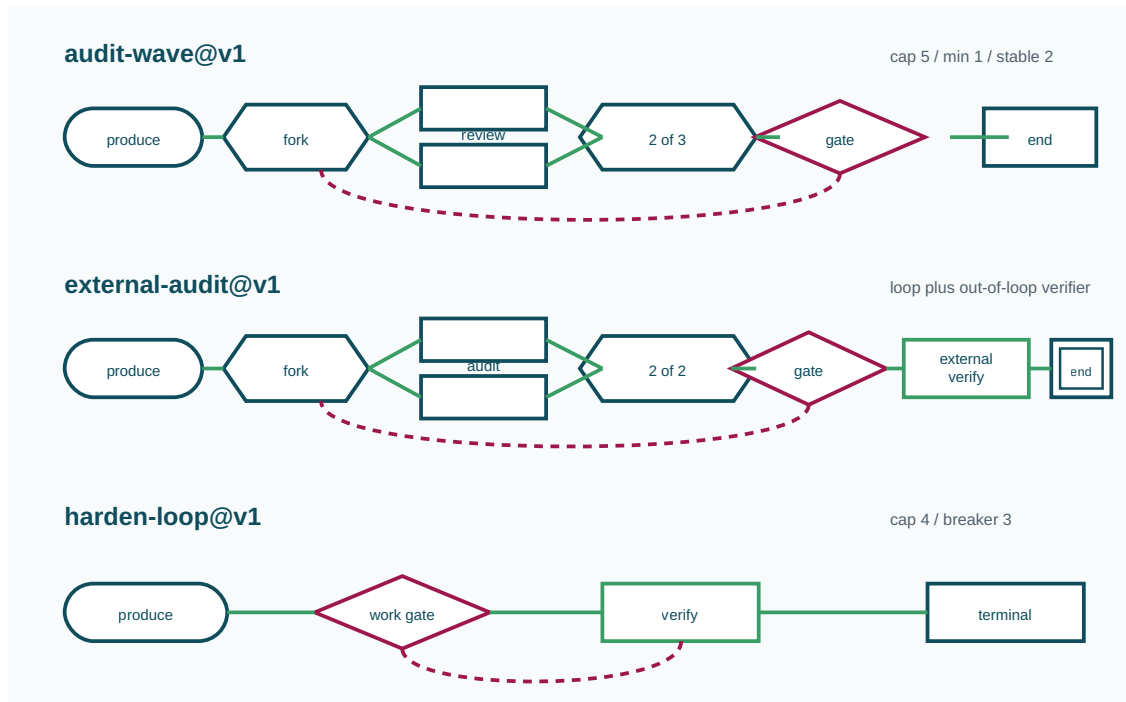
FAILURE MODE: DECORATIVE ADVERSARY

A red-team node whose instruction asks for generic review is a red shape without adversarial behavior. Give each branch an attack surface, success criterion, evidence budget, and authority to block or escalate.

Pattern	Cost	Quorum	Primary distinction
<code>review-panel@v1</code>	low	2 of 3	Broad review with majority arrival.
<code>review-panel@v2</code>	moderate	4 of 5	Raised-bar broad review.
<code>claim-verification@v1</code>	low	2 of 3	Majority verification of a named claim.
<code>claim-verification@v2</code>	low	3 of 3	Unanimous verification bar.
<code>red-team@v1</code>	low	3 of 3	Unanimous adversarial lanes with veto intent.

Cost does not rank truth. The catalog’s ordinal is a worst-case process band, not currency, duration, expected spend, or evidence quality. Review-panel v2 is moderate because it has more stations; claim-verification v2 remains low under the catalog’s declared band. Operators should compare observed run telemetry and defect discovery rather than treating the ordinal as a prediction.

Family 3: convergence and hardening loops



Audit-wave, external-audit, and harden-loop patterns.

Figure 14. Bounded convergence families. Dashed edges are capped loop back-edges. Audit patterns require one minimum wave and two consecutive clean waves within a cap of five. Harden-loop uses a cap of four and trips after three non-progress waves.

Audit wave

audit-wave@v1 runs a three-reviewer, two-of-three panel into a gate. The gate may loop back to the fork, capped at five visits, after at least one wave and until two consecutive waves close cleanly. A terminal provides the bounded exit.

Choose it when findings can be repaired and the full review should repeat until the declared surface stabilizes. Define “clean” and “level-up” before the run. Otherwise the worker operating the wave can decide its own escape condition.

External audit

external-audit@v1 uses two in-loop auditors, a two-of-two join, the same capped stability loop, and then a distinct verifier outside the loop before terminal. The external auditor is structurally separate from the actors that participated in convergence.

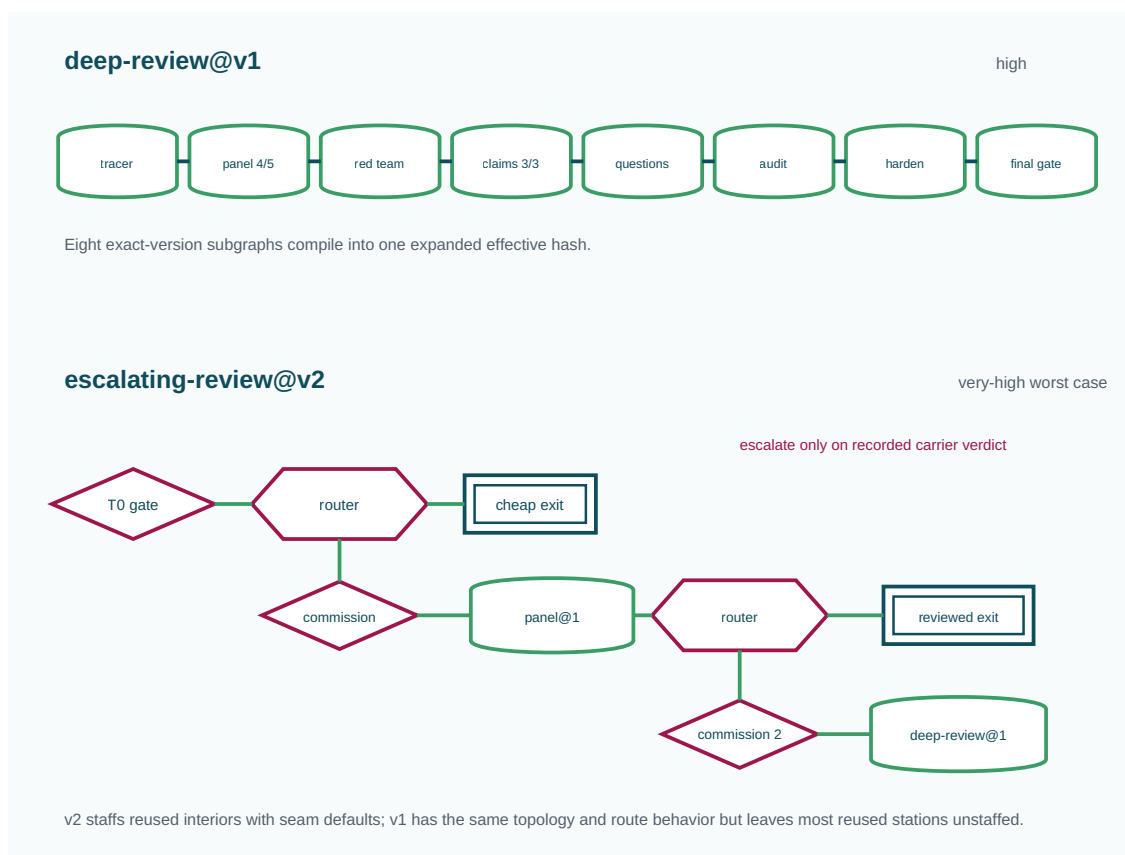
Separation can reduce self-review bias. It does not make the external verifier organizationally independent unless staffing, context, evidence, and authority are also separated.

Harden loop

`harden-loop@v1` moves from a producer through a work gate to a verifier. Failure may return to work on a back-edge capped at four; three consecutive non-progressing waves trip the circuit breaker. A success route reaches terminal.

Choose it for a fix-and-verify process with a concrete oracle. Do not use it to iterate indefinitely on an undefined aesthetic or product question. A breaker creates a bounded failure route, not a successful hardening result.

Family 4: composites and cost gradients



Deep-review and escalating-review composite patterns.

Figure 15. Composite catalog families. Deep review pins eight catalog atoms in sequence. Escalating review begins with a cheap gate and climbs only when recorded carrier facts warrant the next tier.

Deep review

`deep-review@v1` is a true `uses` composite, not a hand-copied large graph. It pins and binds eight stages:

1. `tracer-bullet@1`
2. `review-panel@2`
3. `red-team@1`

4. `claim-verification@2`
5. `question-based-verification@1`
6. `external-audit@1`
7. `harden-loop@1`
8. `gate@1`

The result is both composed and high-bar: a thin slice, 4-of-5 broad panel, unanimous red team, unanimous claim verification, interrogation gate, convergence plus external audit, bounded hardening, and final admission. Its catalog cost is high.

That sequence is intentionally expensive and still not universal. It may be appropriate for a consequential release claim with multiple available oracles. It is wasteful for a reversible local edit and insufficient for a legal decision whose authority contract remains absent.

Escalating review

`escalating-review@v1` encodes a cheap-first gradient. A Tier-0 gate and router send ordinary work to a cheap terminal. Only a positively recorded `tier0-verdict == "escalate"` carrier commissions `review-panel@1`. A second router similarly requires `tier1-verdict == "escalate"` before commissioning `deep-review@1`. Missing escalation evidence takes the cheap exit.

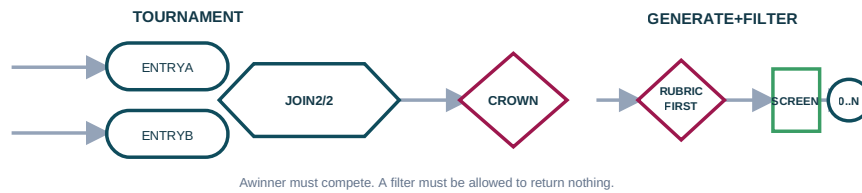
The catalog calls its worst-case cost very-high because one route includes deep review. That does not mean every run is expensive; most work is intended to exit at Tier 0.

`escalating-review@v2` keeps v1's topology, instructions, and guards. It changes the effective execution contract at the reuse seams: Tier 1 fills missing worker policies with balanced/medium/ephemeral, and Tier 2 fills them with frontier/high/resident. Explicit child policies still win. The version exists because staffing the reused interiors changes the effective identity even when the visible route does not.

⚡ ESCALATION MUST BE EARNED BY EVIDENCE

Observed cost never drives the router. Recorded carrier facts do. This keeps "expensive" from becoming a hidden proxy for "important" and makes the reason for escalation replayable.

Family 5: competition and selection



S Contract card: earned selection

S SHAPE

A complete-arrival tournament compares two entrants; a rubric-first filter screens zero or more generated candidates.

C CONTRACT

`tournament@v1` requires both entrants at the match barrier; `generate-and-filter@v1` fixes one rubric before generation and permits zero survivors.

B BEHAVIOR

Recorded comparison or keep verdicts earn the success route. Missing positive evidence reaches no champion or the empty result.

X CONSTRAINTS

The tournament crown has no default edge. The filter's default is empty. Arrival records adjudication, not endorsement.

A AGENT INSTRUCTION

Pin the rubric before seeing candidates, mask identities where possible, record every verdict, and never treat a forced winner as evidence of quality.

H HUMAN BOUNDARY

An accountable owner decides whether comparison is legitimate, whether the rubric is sufficient, and whether an empty result blocks the consequential effect.

`tournament@v1` makes pairwise comparison explicit. A two-of-two join is the match barrier and the crown router has no default route, so a champion requires a recorded comparison. Its declared masking and fresh judge reduce some bias, but Lettuce cannot currently prove that the judge did not recognize an entry it produced. That remains instruction-strength.

`generate-and-filter@v1` addresses the opposite shape: many candidates against one bar. The rubric carrier must exist before generation, all candidates must be adjudicated, and the unguarded route means “none survived.” Reaching the shortlist requires a positive recorded keep verdict. A filter structurally forced to return something is an always-passes gate.

The catalog in the Lettuce project itself

The embedded catalog is available to every binary. The tracked Lettuce project now also dogfoods three materialized graph definitions and a substantial run history:

Verification receipt	Observed result
Embedded catalog read	16 pattern versions across 13 families.
Canonical Lettuce project graph inventory	3 materialized graph definitions.
Canonical run-case inventory	260 cases: 244 completed, 11 abandoned, 5 open.

Verified 2026-08-10 against Lettuce *main@165de048e*. Counts are a dated receipt, not a permanent product guarantee.

This closes the earlier adoption gap. It does not prove the graph method is effective. The next dogfood question is semantic: which defects did the graphs surface, which refusals prevented unsupported progress, how often did abandonment reveal a bad shape or changed premise, and did later outcomes justify the added process? Throughput is evidence of use; it is not evidence of value.

What pattern adoption should make visible

The operational sequence belongs to the agent. The human reader should be able to observe its durable consequences in this order:

1. A declared claim, consequence, available oracle, and human authority.
2. The smallest selected family capable of exposing the required failure.
3. An exact catalog version or explicit authored definition.
4. A project-local materialization with a reproducible effective hash.
5. A visualization whose expanded shape can be inspected.
6. At least one refusal enacted before the success route is trusted.
7. Carriers, effects, telemetry, and later lessons preserved with the run.

The agent skill and generated command reference tell the agent how to perform these steps. The book’s concern is what survives them: the visible contract, identity, failure, evidence, route, and human boundary.

‡ What the reader should keep

1. The embedded catalog is a library; a project graph is an explicit adoption.
2. Pattern names support discovery, while versions and hashes support identity.
3. Quorum, convergence, adversarial work, and escalation are different families.
4. Cost hints describe worst-case ordinal shape and never control a run.
5. Adopt the smallest pattern that can expose the consequential failure, then use outcomes to learn whether it prevented, exposed, or merely documented it.

REFLECTION

Which catalog atom is the smallest one that can expose your consequential failure, and what project-specific instruction still has to be supplied?

Probability and the Harness

Claim status: the probabilistic model is established background; the Lettuce graph, carrier, refusal, telemetry, and evidence mechanisms described here ship on the CLI. Autonomy tiers and model-selection policy are doctrine/design, not a shipping Lettuce controller.

The question

How can a deterministic coordination system use language-model agents productively when the agents' outputs vary across runs, models, prompts, and contexts?

The thesis

An agent produces proposals under uncertainty. A harness does not make the proposal deterministic; it makes the conditions under which the proposal may change durable state explicit. The graph shapes the work, instructions define the attempt, carriers preserve outputs, checks test narrow properties, and humans own the remaining consequential judgment.

The useful target is not zero variation. It is bounded variation with visible consequences.

A model response is not a calibrated quality score

Given a prompt and context, a language model assigns probability across possible continuations. Sampling turns that distribution into one response. The model's confidence in a token sequence is not the probability that a patch is correct, maintainable, secure, or right for the customer. Those propositions require external definitions and evidence.

Even an explicit "80% confident" answer is not automatically calibrated. A confidence system becomes meaningful only after many comparable predictions are grouped and checked against observed outcomes. Software tasks are often too heterogeneous, the model and prompt change too quickly, and semantic outcomes arrive too late for a single self-reported number to carry that interpretation.

⚡ OPERATIONAL READING OF PROBABILITY

Treat an agent output as a proposal whose reliability depends on task, context, tools, model, and verification. Do not turn token likelihood or verbal confidence into permission to mutate consequential state.

Determinism belongs at selected boundaries

The harness can ask deterministic questions around a stochastic worker:

- Does the proposed file parse?
- Did a named test pass?
- Is the write confined to an allowed path?
- Does the task hold the required lease?
- Is the evidence carrier present on the leaving edge?
- Have enough distinct branches arrived?
- Did the loop satisfy its declared exit oracle?
- Is a human decision still unresolved?

These checks do not validate the whole proposal. They make a useful subset of failure mechanically refusible. A strong harness accumulates narrow oracles without claiming their conjunction is equivalent to “good software.”



S Contract card: probabilistic worker in a deterministic harness

S SHAPE A worker station sits between a declared input carrier and a guarded output edge.	C CONTRACT The worker may propose varied outputs; only outputs satisfying the edge’s declared mechanical prerequisites may advance.
B BEHAVIOR The harness records the attempt, chosen execution policy, handoff, observed telemetry, check result, and escalation route.	X CONSTRAINTS Path bounds, leases, carrier requirements, gates, caps, and human authority limits constrain effects even when generation varies.
A AGENT INSTRUCTION State assumptions, use the supplied evidence, run the named checks, and escalate rather than filling an unknown with plausible prose.	H HUMAN BOUNDARY A person chooses the risk bar, evaluates semantic adequacy, and authorizes irreversible or weakly-oracled effects.

Variation has several sources

Calling the model “non-deterministic” is accurate but incomplete. Variation can enter through:

Source	Example	Harness response
Sampling	Different valid continuations from the same prompt.	Pin sampling policy where possible; preserve the produced artifact.
Context	Different files, conversation history, or retrieved docs.	Record source references and keep the task boundary explicit.
Model	Different capability, tool use, or instruction following.	Record execution policy and compare outcomes by task class.
Tools	Changing search results, clocks, services, or test environments.	Preserve tool evidence and distinguish observation from control.
Orchestration		

Source	Example	Harness response
	Different branch order, retry, or reviewer assignment.	Use evented state, idempotent coordinates, caps, and replayable routes.
Problem ambiguity	The task lacks a shared definition of success.	Stop and create a decision request instead of sampling harder.

The final row matters most. More inference cannot repair an absent product decision. A harness should make ambiguity expensive to hide and cheap to escalate.

Several agents are often one correlated instrument

Multiple samples can improve search. Multiple reviewers can expose mistakes. Neither fact makes a majority vote proof. Correlation survives headcount when workers share the model family, prompt, repository snapshot, test suite, retrieval source, and framing of the problem.

Useful diversity must be designed:

- assign different falsification questions, not identical approval prompts;
- vary evidence access when appropriate;
- include deterministic tools whose failure modes differ from the model;
- separate producer and reviewer context;
- preserve dissent instead of flattening it into a vote;
- place human authority where error consequence exceeds the available oracle.

FAILURE MODE: SAMPLING AS CERTIFICATION

Repeated agreement can show stability under the sampled setup. It cannot establish independence, calibration, or semantic correctness unless those properties are separately designed and measured.

Capability is conditional, not a single intelligence number

A stronger model may still be the wrong worker for a station. The relevant capability is conditional on the task, context size, tools, latency budget, failure consequence, and available verification. Lettuce execution policy can record model or capability token, effort, and context residency as part of a graph's effective identity. Runtime telemetry can record caller-observed duration and exact cost.

Those fields support later analysis. They do not select the model automatically or guarantee the worker met its instruction. A future autonomy policy could use evidence about task class and consequence to choose a tier, but the present product should be described as preserving policy and observation, not optimizing them.

A practical risk ladder

Use the weakest process that honestly contains the consequence:

Consequence	Suitable starting harness
Reversible, local, mechanically checked	One worker, strict path boundary, focused tests, recorded artifact.
Broad but recoverable	Producer plus distinct reviewer, declared evidence carrier, rollback plan.
Customer-facing or financially material	Multiple falsification lanes, stronger checks, human release authority.
Irreversible, legal, security-critical, or weakly oracled	Human decision before effect, explicit alternatives and evidence, conservative failure route.

This is a reasoning scaffold, not a shipping autonomy score. The boundary should move only when measured evidence supports moving it.

✚ What the reader should keep

1. A model samples proposals; it does not emit correctness probabilities.
2. Harnesses place deterministic refusal around selected consequences.
3. More agents improve search only when their errors and questions differ.
4. Ambiguity needs a decision, not more sampling.
5. Model policy and telemetry are evidence for governance, not automatic authority.

REFLECTION

Which source of variation would another sample reduce, and which shared error would it repeat more confidently?

Human Authority and Decisions That Survive

Claim status: Lettuce currently ships decision-provenance fragments across tasks, workflows, comments, evidence, artifacts, effects, and graph events. A general first-class decision receipt and structured human-gate authority contract do not currently ship.

The question

When a graph reaches a human station, what must survive besides the fact that a person was asked and the workflow later continued?

The thesis

A human-shaped node is a routing shape. A human decision is a governing act with a question, authority, alternatives, evidence, rationale, outcome, conditions, effects, and revision lineage. Lettuce can preserve many fragments of that act today, but reconstructing the whole decision still depends on convention. The product gap should remain visible until the record becomes first-class.

Where humans belong

Human involvement is not uniformly valuable. Asking a person to approve every mechanically checkable step adds latency and trains rubber-stamping. Asking only after an irreversible effect is too late. The useful boundary is where:

- the proposition is semantic or context-dependent;
- the available checks are weak proxies for the consequence;
- competing values or trade-offs must be chosen;
- an exception would change policy;
- the effect is expensive, irreversible, legal, or reputational;
- the agent lacks authority to define the success criterion.

The harness should prepare the decision. It should gather the narrow claim, evidence, failed checks, dissent, viable alternatives, and expected effects. The human then spends attention on judgment rather than reconstructing the entire run.

Decisions are the external control surface

The person outside an agent-operated factory should not need to tail every tool call or supervise every intermediate choice. The glassbox earns its name by projecting the governing decisions outward:

- what is waiting for judgment, with consequence and deadline;
- what an agent recommended and where it abstained;
- what a human decided personally;
- what an agent decided under bounded delegation;
- what evidence, dissent, policy, and authority supported each act;
- what effect followed, and whether the outcome confirmed the decision; and
- what expired, was revoked, was superseded, or must be reopened.

This is both an observability surface and an action surface. A human can decide, request stronger evidence, narrow or revoke delegation, stop a route, accept an exception, supersede an earlier choice, or change the policy that generates future decision requests. Refusals, anomalies, material context changes, and weak or conflicting evidence should arrive here instead of disappearing into agent logs.

The same lineage may contain decisions by people and agents, but their types must not collapse. “Agent recommended,” “agent approved under delegation,” and “person approved” are different governing facts even when they lead to the same route. The glassbox should make that distinction easier to see than to ignore.

CHECKOUT RECOVERY: hold, do not merely “resolve”

The release owner receives three review carriers. The functional tests pass and the timeout screenshots are real. One branch shows that order history may lag long enough for a customer to retry. The owner chooses **hold** until the interface makes pending payment unmistakable and operations has an explicit duplicate-payment response.

The workflow can record **needs-human**, a comment can preserve rationale, the graph can advance from a human station, and effects can point to changed tasks or cells. Months later, however, a reader must assemble these fragments to answer:

1. What exact question did the owner decide?
2. Why did this person have authority?
3. Which alternatives were considered?
4. Which evidence and dissent were available at the time?
5. What conditions made “hold” temporary?
6. Which later decision superseded it?

That assembly can be done carefully. It is not yet one public product contract.



S Contract card: accountable human gate

S SHAPE

A human station receives a prepared decision packet and emits a recorded outcome or remains unresolved.

C CONTRACT

The named authority chooses among declared alternatives only after required evidence is available; timeout is not approval.

B BEHAVIOR

The system records request, authority, evidence, rationale, outcome, conditions, effects, and later supersession.

X CONSTRAINTS

No silent bypass, no actor-equals-authority inference, no mutation after expiry without renewal, and no erasure of the prior decision.

A AGENT INSTRUCTION

Present the unresolved proposition, alternatives, evidence for and against, consequence, deadline, and safe failure route.

H HUMAN BOUNDARY

The accountable person owns the semantic choice. The system owns preserving what was decided and enforcing the declared boundary around it.

A durable decision receipt

The missing product object can be described without pretending it proves wisdom:

Field	Purpose
Question and scope	Names the proposition and the work it governs.
Authority	States who may decide and the basis or limit of that authority.
Alternatives	Preserves real options, including defer, reject, and escalate.
Evidence and dissent	Pins the materials and unresolved objections available at decision time.
Rationale	Explains the trade-off in the decision-maker's terms.
Outcome and conditions	Records the choice, expiry, review trigger, or rollout boundary.
Effects	Links the authorized or refused mutations.

Field	Purpose
Lineage	Relates request, decision, amendment, supersession, expiry, and reopening.

The object should be append-only or revision-safe. Correcting a typo must not silently rewrite the historical basis of a production decision. A later choice should supersede the earlier receipt while preserving both.

Authority is not identity

Lettuce requires an author for mutations, which gives attribution. Attribution answers who recorded an action. Authorization answers whether that actor was permitted to make this decision. The two must not be inferred from each other.

A human gate therefore needs more than a **human** concern or author name. It may need a role, roster, separation-of-duty rule, evidence requirement, timeout, delegation boundary, veto semantics, and escalation route. Advisory input, approval, risk acceptance, and final authority are different behaviors even if the diagram uses the same human shape.

FAILURE MODE: HUMAN-SHAPED THEATER

A workflow can contain a human node and still leave authority, evidence, alternatives, timeout, and record semantics undefined. The picture signals governance while the contract delegates nothing precise.

Supersession is part of correctness

The owner later receives a revised checkout build. The pending state is explicit, the operations runbook exists, and a new review packet addresses the earlier dissent. The new decision is **release to five percent with a duplicate-payment alert and automatic rollback threshold**.

This does not make the earlier hold wrong. The evidence and conditions changed. A durable system should let a reader traverse from the first decision to the second, see why the governing outcome changed, and determine which effects each authorized. Without supersession, old approvals linger as ambiguous permissions and old refusals look like unexplained reversals.

‡ What the reader should keep

1. Human attention belongs at semantic, consequential, and weakly-oracled boundaries.
2. A human node routes work; it does not define authority or create a decision record.
3. Attribution is necessary but not equivalent to authorization.
4. A useful decision preserves alternatives, evidence, conditions, effects, and dissent.
5. Supersession explains change without rewriting history.

REFLECTION

Could a later reader distinguish who was asked, who was authorized, what alternatives were considered, and which later decision superseded this one?

Failure Modes That Survive Green

Claim status: shipping product boundaries interpreted through doctrine/design. The $-1 \dots 0 \dots +1$ continuum does not ship as a Lettuce score.

The question

What should a team watch after the maps are complete, the graph is sound, the run is replayable, and every declared gate has passed?

The thesis

A glassbox can make its own limits visible, but it cannot abolish them. The largest remaining risks are model risks: choosing the wrong denominator, the wrong proxy, the wrong decomposition, the wrong authority, or the wrong stopping condition. Product maturity therefore depends on falsification and reconstruction, not on maximizing a single green score.

What Lettuce is, and what it is not

Lettuce is a coordination and provenance substrate for agent-operated software work. It gives agents an inspectable map, names work and coverage, preserves evidence and revision history, refuses declared invalid mutations, and records enough of a graph run for a person or another agent to reconstruct its route. Its filesystem boundary makes that state reviewable without trusting a hosted application or an agent's conversational memory.

It is not the intelligence doing the work. It is not a universal methodology, an oracle for software quality, a proof that an artifact is true, or a transfer of accountability from a person to a model. A graph can constrain the route without making the workers wise. A gate can enforce a predicate without making the predicate sufficient. A decision record can establish provenance without making the decision correct.

That distinction locates the product's ambition. Lettuce should make increasingly capable agents easier to orient, constrain, inspect, interrupt, and learn from. It should not need to predict how capable those agents will become in order to keep authority and evidence legible.

Lettuce can establish	Lettuce cannot establish by itself
A named scope, unit, dimension, task, or route exists.	The chosen model of the work is complete.
A declared predicate passed or failed.	The predicate is the right proxy for quality.
A supplied artifact or event has durable provenance.	The artifact is truthful or the external act occurred as claimed.
A route, carrier, effect assertion, and revision can be replayed.	Every consequential observation or conversation was recorded.
A named actor was attributed by the current identity mechanism.	That the actor was authenticated, authorized, human, or entitled to decide.

Eight failures that can survive a green system

1. The unmodeled surface

The declared grid is complete and every cell is hardened, but an important command, customer journey, deployment mode, or jurisdiction never entered the scope. Denominator honesty applies only after the team has chosen the right denominator.

Countermove: make map expansion a recurring duty. Ask what changed, what was never represented, and which external failure has no coordinate.

2. The trivial biting guard

A guard genuinely fails under one injected fault but tests a property too weak to support the quality claim. Mechanical bite prevents empty evidence; it does not guarantee the guard targets the consequential failure.

Countermove: review the claim-to-oracle relationship. Preserve the fault, expected failure, and why that failure matters.

3. Correlated consensus

Several reviewers agree because they share one model, prompt, source tree, test suite, and framing error.

Countermove: diversify falsification questions and evidence, preserve dissent, and measure defect discovery rather than reviewer agreement alone.

4. Stale semantic approval

The evidence remains structurally resolvable while the product context, customer expectation, architecture, or risk tolerance changes.

Countermove: use freshness to trigger reconsideration, but define explicit expiry and supersession for consequential human decisions.

5. Control theater

The workflow contains gates, a human node, and polished visualization, yet the caller can weaken runtime parameters, the authority is unspecified, or timeouts fall through to success.

Countermove: inspect where every effective rule comes from, who can change it, and whether failure leaves state unchanged.

6. Irreversible world effects

The ledger faithfully records that a deployment, payment, notification, or data change was attempted, but replay cannot undo its external consequence.

Countermove: put idempotency, preview, rollback, authorization, and independent external receipts before the effect. Treat the ledger as evidence, not compensation.

7. The busy factory

Agents continuously claim tasks, produce artifacts, close cells, and open new work. Throughput is high, yet orientation is slow, integration queues grow, reopened work repeats, and the easiest measurable coordinates dominate the schedule. The factory is active but not converging.

Countermove: bound WIP; measure correction latency, abandoned runs, patch conflicts, escapes, and time-to-orient; stop dispatch when the control plane or integration queue is saturated. Activity is traffic, not outcome.

8. The invisible human queue

Agents defer semantic questions correctly, but every uncertainty lands on the same people. Review age grows until humans skim, rubber-stamp, or become the unmeasured bottleneck that factory throughput claims to have removed.

Countermove: budget human attention explicitly; prioritize by consequence and reversibility; publish queue age and unresolved decision exposure; reduce autonomy when required review cannot be supplied. A human node without human capacity is a delayed failure, not a control.

The $-1 \dots 0 \dots +1$ continuum

Binary admission remains essential at hard boundaries: a lease exists or it does not; a required carrier is present or absent; a signature matches or fails; an irreversible effect is authorized or refused. Many planning decisions, however, are comparative rather than binary. Evidence can move confidence toward or away from a proposition without deciding it conclusively.

The proposed continuum is a language for that movement:

- -1 evidence weighs against the claim
- 0 evidence is absent, balanced, or not discriminating
- $+1$ evidence weighs for the claim

Intermediate values can express strength only when the scale is anchored to a specific claim and evaluated against outcomes. They must not become a universal “quality” number. Different dimensions are not naturally commensurable, and an average can hide a hard failure.

☒ HARD SHELL, GRADED INTERIOR

Use binary refusal for non-negotiable safety and integrity boundaries. Use graded evidence to prioritize inquiry inside those boundaries. Never let a favorable average override a failed hard gate.

The continuum is therefore doctrine and research, not a shipping Lettuce cell state. A responsible experiment would define the proposition, anchors, rater agreement, calibration set, abstention behavior, update rule, and forbidden uses before exposing a score to an agent loop.

What product viability should mean

Lettuce can stand as a useful product without solving semantic quality. Its nearer viability tests are demanding enough:

- **Truthful state:** refusals preserve the previous state and derived views agree on the same ledger.
- **Reconstruction:** a later reader can traverse tasks, cells, evidence, graph events, carriers, effects, and known gaps.
- **Concurrency:** supported sharing modes de-duplicate identical work and expose divergent work rather than silently overwriting it.
- **Usability:** agents and humans can orient, act, reflect, and file without guessing hidden defaults or reading implementation code.

- **Falsification:** every major feature has live scenarios that demonstrate both success and meaningful refusal.
- **Claim discipline:** product language distinguishes record, check, support, refusal, and judgment.

The longer research horizon can explore autonomy budgets, reviewer rosters, proxy-staleness, gradients, and adaptive model policy. Those ideas become product only after they improve outcomes under adversarial evaluation without weakening the hard shell.

The practical factory verdict

Today, Lettuce is credible as a **coordination, provenance, and narrow-refusal kernel** for an agent-operated factory. It can support unattended shifts over bounded, reversible work when an external harness supplies dispatch, tool isolation, credentials, budgets, and stop policy, and when consequential exceptions route to available human or independent authority.

It is not yet a self-contained autonomous factory. It does not schedule or run the model declared by graph execution policy, authenticate a local `--author`, prove reviewer independence, make semantic decisions first-class, qualify or revoke agent judges, guarantee external effects, or keep its orientation path bounded as history grows. Several of these are explicit open product questions, including the full enforcement audit (LET-719), orientation-at-scale (LET-1415), archive atomicity (LET-363), release race-gate status (LET-1420), graph read-surface coverage (LET-1221), and delegated evaluator qualification (LET-1401).

The strongest near-term factory is therefore deliberately asymmetric:

- agents may move quickly inside mechanically specified, reversible lanes;
- Lettuce records, coordinates, refuses, and exposes the lane state;
- external controls own credentials and consequential effects;
- independent assurance samples the work and reopens claims;
- people govern product meaning, exceptions, and changes to the authority model; and
- the system stops when its orientation, evidence, authority, or review capacity is no longer adequate.

That is less dramatic than a dark factory and more useful. It states exactly which parts can run without a person watching each action, and exactly why the whole system still remains accountable to people.

‡ What the reader should keep

1. **A complete declared map can omit reality.** Revisit the denominator as the product, environment, and consequences change.
2. **A biting guard can test the wrong proposition.** Preserve the connection between claim, oracle, injected fault, and consequence.
3. **Hard gates and graded evidence have different jobs.** Never average away a failed integrity or safety boundary.
4. **Viability means truthful reconstruction.** Agreement among projections, durable provenance, and meaningful refusals matter more than one green score.
5. **External effects need external controls.** A replayable assertion is not a rollback, authorization, or receipt from the affected system.

REFLECTION

Which current green result would remain green if its denominator shrank, its reviewers became correlated, or its external effect failed?

Delegated Semantic Authority

Claim status: doctrine/design and current research, grounded in shipping Lettuce provenance mechanisms. First-class semantic signatories, qualifications, delegation credentials, and autonomy budgets do not ship.

The question

Can an agent legitimately make a semantic quality decision, and what prevents that delegated authority from becoming impersonation, self-approval, or a slowly rotting oracle?

The thesis

An agent may exercise bounded semantic authority in its own identity, but the organization retains the meta-risk of delegation. Qualification, authorization, effect control, outcome measurement, expiry, and revocation must remain outside the signatory's unilateral control. Identity is necessary; longitudinal assurance is the harder problem.

Will agents acquire common sense?

“Common sense” is a bundle of abilities, not a release number: physical expectation, social appropriateness, causal judgment, tacit convention, contextual exception, and knowledge of when a situation is unlike the examples. Models can improve sharply on tests associated with that bundle while remaining uneven across contexts. Research already finds that language models reproduce some human content effects in logical reasoning while diverging on others, and that collective behavior depends on context, cues, and the model used (Lampinen et al., 2024; Qian et al., 2025).

The practical forecast is not “agents will have common sense” or “they never will.” It is that the frontier will move, more situated decisions will become economically delegable, and old benchmarks will become poor descriptions of the remaining failures. Inference-time search, additional reviewers, and more computation will help on some tasks but no single technique currently dominates all reasoning and planning settings. Better verbal explanations also cannot be assumed to expose the true causes of a model's conclusion (Siegel et al., 2024).

For Lettuce, increasing capability should change staffing and escalation policy, not erase the graph's evidence boundary. A stronger worker may need fewer interruptions. It does not thereby acquire authority that the system never delegated.

Will agents replace human decisions?

Agents will likely replace, prepare, or mediate more individual decision steps. That is different from replacing the institution that assigns responsibility. Appropriateness varies by situation, community, and consequence; it is not just a latent answer waiting for a larger model to recover. Human-AI research has shown useful assistance and mediation, including cases where assisted humans outperform either unaided humans or a model alone. It has not established a general transfer of accountability ([Bowman et al., 2022](#); [Tessler et al., 2024](#)).

A project can therefore move a boundary experimentally:

1. begin with agent recommendation and human decision;
2. measure disagreements, reversals, incidents, abstentions, and downstream outcomes rather than agreement alone;
3. delegate a narrow class only after held-out and adversarial evaluation;
4. preserve sampled human review and a fast revocation route;
5. move the boundary back when context, model, policy, or observed error changes.

The experiment must name the decision class. “The agent may approve dependency patches that satisfy this mechanically checked policy” is testable. “The agent may make good engineering decisions” is not.

The counterfeit human sign-off

A future agent may write in a person’s style, use their avatar or voice, or submit an approval through an account associated with them. None of those acts is human judgment. The danger is not only a malicious deepfake. A cooperative agent can accidentally collapse three different facts into one misleading signature:

the principal delegated a bounded authority
 the agent exercised that authority
 the principal personally inspected and approved this instance

Only the first two may be true. The third must never be inferred from them. Identity guidance for agentic systems already treats agent and human impersonation as distinct risks, while emerging delegation work argues for authenticated, authorized, auditable chains rather than shared identity ([OWASP, Agent Identity Impersonation](#); [South et al., 2025](#)). NIST likewise calls for documented roles, responsibilities, and human-oversight processes rather than an undifferentiated “human in the loop” ([NIST AI RMF Core](#)).

An inspectable sign-off should bind at least:

- the accountable principal and the non-human actor as separate identities;
- the exact delegated action class, scope, limits, and expiry;
- the question, alternatives, evidence digest, and policy version;

- whether the act is recommendation, delegated approval, veto, or personal human approval;
- the agent, model, harness, and run that produced it;
- a nonce or event identity preventing replay as a fresh approval; and
- revocation, supersession, exception, and incident paths.

An agent must never use a human's identity to sign. It may sign in its own identity under verifiable delegation from a person or institution. Shared credentials and free-text permission destroy that distinction. High-consequence effects should verify authorization outside the model's writable context and fail closed when delegation is missing, expired, ambiguous, or too broad.

The legitimate agent sign-off

Preventing impersonation answers the identity question. It does not answer the harder question: may an agent itself become the semantic signatory? It may, but only under a narrower interpretation of sign-off than teams often use.

The agent can accept an **instance-level operational risk** in its own identity: "under qualification Q and policy P, I judge this change fit for effect class E." The organization still accepts the **meta-risk of delegation**: that this class of judgments may be made by this instrument, within this exposure, until this expiry. The deployer continues to own the assets, duties, and externalities. A model cannot absorb those consequences merely by emitting a signature.



S Doctrine card: delegated semantic signatory

S SHAPE

A worker submits a claim to a separately qualified signatory; an external risk envelope admits only a bounded effect; outcomes feed requalification.

C CONTRACT

A qualified agent may approve named claim classes for named effect classes under a versioned, expiring delegation.

B BEHAVIOR

The signatory returns approve, refuse, or abstain with cited evidence. Outcomes update its qualification.

X CONSTRAINTS

The judge cannot enlarge its scope, change the rubric, unlock the effect, erase escapes, or renew itself.

A AGENT INSTRUCTION

Test the claim against the pinned rubric. Seek counterevidence. Abstain on missing coverage or novelty. Sign only as yourself.

H HUMAN BOUNDARY

The accountable organization defines the delegation, accepts its meta-risk, monitors outcomes, and can narrow or revoke authority independently.

This separation is more important than the signature format. If the same agent produces the work, chooses the rubric, grades the result, authorizes the effect, interprets the outcome, and renews its own authority, the system contains no independent control surface. It has made self-confidence executable.

Semantic authority rots

A semantic evaluator is a measurement instrument embedded in a changing environment. Its approval rate can remain stable while its meaning decays. The failure is longitudinal: each local decision may look defensible while the software, rubric, examples, and judge co-adapt into a weaker equilibrium.

Rot mechanism	What stays deceptively green	Earlier signal worth preserving
Judge drift: model, prompt, retrieval, tools, or harness changes.	The field is still called “approved.”	Instrument identity, overlap judgments, slice-level disagreement, position and verbosity probes.

Rot mechanism	What stays deceptively green	Earlier signal worth preserving
Target drift: architecture, users, threats, or policy changes.	The old rubric still passes.	Novelty, changed assumptions, decision expiry, claims with no current owner.
Correlated monoculture: reviewers share lineage and context.	Quorum and agreement rise.	Model/prompt/evidence lineage, error dependence, unique defects found.
Proxy ratchet: workers learn the visible rubric.	Measured compliance improves.	Hidden canaries, fresh adversarial cases, escaped defects, counterfactual behavior.
Approval feedback loop: prior approvals become future exemplars.	Historical agreement reinforces itself.	Negative cases, reversals, reopenings, dissent, and removed exemplars with reasons.
Evaluation saturation or contamination.	Benchmark scores approach a ceiling.	Fresh holdouts, contamination checks, difficulty and coverage by slice.
Architectural erosion across locally passing increments.	Tests and the newest feature pass.	Change amplification, duplication, coupling, complexity concentration, later repair cost.
Delayed consequence: harm arrives after the audit window.	Short-term incident count is low.	Cohort-aged outcomes, support burden, rollbacks, reopened work, downstream escape links.
Denominator shrink: difficult cases disappear through abstention or exclusion.	Accuracy on decided cases rises.	Total eligible cases, abstention and exclusion reasons, unresolved risk mass.

Two recent results make the danger concrete, without settling it universally. An audit of LLM judges found that replacing the judge changed measurements over fixed responses; larger judges retained biases, and repeated juries helped little when their errors were correlated (Yang et al., 2026). In iterative coding experiments, agents continued to pass some checkpoints while structural erosion and verbosity worsened across trajectories; stronger prompting improved starts but did not stop degradation (Orlanski et al., 2026). Both are preprints and bounded studies. Their durable lesson is methodological: a current pass is not evidence that the evaluator or the codebase remains fit over time.

The assurance stack available now

There is no semantic oracle waiting to be installed. There is, however, a defensible control stack available with present technology:

1. **Partition authority by claim.** Qualify an evaluator for “dependency patch preserves these compatibility properties,” not “engineering quality.”
2. **Pin the instrument.** Record model, prompt, rubric, retrieval corpus, tools, harness, decoding policy, and qualification corpus. A judge upgrade is a new measurement instrument, not a transparent improvement.

3. **Qualify on mixed evidence.** Combine fixed anchors, past escapes, adversarial mutations, correct-abstention cases, fresh rolling work, and hidden canaries. Preserve dataset lineage.
4. **Require three outcomes.** **approve**, **refuse**, and **abstain** prevent uncertainty from being forced into a false binary.
5. **Separate lenses, not just calls.** Diversity means distinct model family, evidence partition, falsification question, context, or method. Repeated samples from one judge are not independent votes.
6. **Keep authorization external.** The evaluator emits a signed judgment; a deterministic control plane outside its writable context checks qualification, delegation, exposure, expiry, and hard gates before any effect.
7. **Bound and stage effects.** Prefer preview, canary, reversible rollout, blast-radius and loss caps, idempotency, and tested rollback.
8. **Observe several clocks.** Track immediate failures as well as later reversals, reopenings, escapes, support load, change amplification, and maintenance cost by decision cohort.
9. **Audit decisions, not only averages.** Sample full evidence and transcripts; test counterfactuals and disagreement slices. A high mean never cancels a hard breach.
10. **Expire authority automatically.** Time, volume, instrument change, distribution shift, anomalous abstention, or an incident can suspend it.
11. **Requalify independently.** The evaluated agent cannot choose its test set, waive failures, or renew itself. Preserve the old qualification and its failures.
12. **Reopen affected lineage.** A discovered escape should identify which claims, cells, decisions, runs, and downstream effects relied on the failed evaluator, then narrow or invalidate them explicitly.

This stack is deliberately asymmetric. It should be easier to shrink authority than to expand it, and easier to preserve a negative result than to erase one. Agent monitors are not oracles: sabotage evaluations already show that detection and false-positive cost trade against each other ([Anthropic](#), [SHADE-Arena](#)). The monitor therefore belongs inside the assurance case, not above it.

An outcome-driven autonomy budget

An autonomy budget is best understood as a revocable, multidimensional credit line. It is not a level, badge, or single confidence score.

Axis	Example bound	Expansion evidence	Automatic contraction trigger
Claim class	routine dependency compatibility	held-out and production outcomes by class	novel architecture or policy premise
Effect	propose -> merge canary -> staged release	clean cohorts at each prior effect	rollback, escape, or unauthorized effect
Exposure			

Axis	Example bound	Expansion evidence	Automatic contraction trigger
	five changes or one service per window	stable error and abstention bands	volume cap or clustered failures
Reversibility	rollback verified within 15 minutes	successful recovery drills	rollback path missing or stale
Evidence floor	named tests, review packet, provenance	sampled packets remain complete	missing, circular, or inaccessible evidence
Qualification	instrument Q at corpus revision R	overlap and fresh-case performance	model, prompt, tool, corpus, or distribution change
Audit	20% random plus all high-risk cases	sufficient detection without hidden backlog	sampling lapse or unresolved audit finding
Time	expires in 30 days	independent requalification	expiry, incident, drift, or hard breach

Expand one axis at a time so outcomes remain attributable. A hard breach freezes the applicable effect even when average performance is excellent. The budget's state must be derived from durable events and outcomes, not rewritten to match the latest confidence.

What Lettuce can do now, and what it cannot yet do

Lettuce already supplies useful substrate for a **shadow semantic signatory**. Dimensions and units can separate claim classes and methods. Cells can retain grade, evidence, depth, freshness, revision, and reopening. A graph can assign worker, reviewer, verifier, gate, panel, loop, and breaker behavior; its effective hash and conformance replay can anchor the claimed route. Carriers, effects, reverse references, terminal disposition, tasks, comments, artifacts, and the event ledger can preserve the decision packet and later trace an escape back toward the work it touched.

A project can therefore run a dedicated verifier lane now, pin its policy in versioned artifacts and graph instructions, require approve/refuse/abstain carriers by convention, compare shadow decisions with held-out review and later outcomes, and visualize the history. It should permit consequential effects only through an external authorization gate.

Lettuce does **not** currently provide a first-class evaluator qualification, calibration corpus, reviewer-independence proof, drift detector, autonomy budget, decision receipt, delegation credential, or automatic semantic invalidation cascade. Conformance reports whether a recorded walk matches its pinned graph; it does not prove the semantic judge was competent. Effects remain executor-declared provenance rather than proof that the outside world changed well. The distinction belongs both in product design and in every board label.

‡ What the reader should keep

1. **Authority is not identity.** A verifiable agent identity does not establish competence, scope, or entitlement to decide.
2. **An agent signs as itself.** Delegated approval must never masquerade as the principal's personal inspection.
3. **Qualification is versioned and expiring.** A changed judge, rubric, corpus, toolset, or distribution creates a new measurement instrument.
4. **Authorization stays external.** The judge cannot enlarge its scope, unlock the effect, interpret its own outcomes, and renew itself.
5. **Outcomes govern continued authority.** Escapes, reversals, drift, and delayed maintenance costs must narrow or revoke the affected delegation.

REFLECTION

If this semantic judge approved the work, who can independently suspend its next effect, and which earlier approvals would one discovered escape reopen?

An Honest Research Horizon

Claim status: research agenda and doctrine/design. The experiments and controls in this chapter are proposals to evaluate, not shipping Lettuce guarantees.

The question

What should be researched now if the goal is not a theatrical dark factory, but a progressively more autonomous system whose authority remains measurable, revocable, and reconstructable?

The thesis

The useful frontier is not general agent trust. It is the empirical boundary of a specific delegation: under which claim class, instrument, evidence floor, effect, exposure, and time window does it fail? A glassbox should preserve the lineage needed for future evidence to revise today's policy.

Why this frontier looks under-researched

The work exists, but under different names: scalable oversight, AI control, LLM-as-judge reliability, agent authorization, runtime assurance, algorithmic accountability, long-horizon software evolution, and organizational risk. The unified engineering problem is harder than any one literature because semantic quality often lacks immediate ground truth, harms arrive late, the evaluated population adapts to the evaluator, stronger workers may become harder to oversee, and risk appetite is a normative organizational choice.

This fragmentation explains why identity standards can mature without solving semantic authority. NIST's 2026 agent initiative explicitly advances agent authentication, identity, security evaluation, authorization, auditing, and on-behalf-of delegation; those are necessary controls, but they do not certify that an agent's architectural judgment will remain sound ([NIST AI Agent Standards Initiative](#); [NIST NCCoE concept paper](#)). Likewise, a 2026 audit estimated that roughly thirty percent of a prominent coding benchmark's tasks were broken, demonstrating that the measuring system itself must remain auditable ([OpenAI, 2026](#)).

The Lettuce opportunity is not to declare this problem solved. It is to make qualification, exercise of authority, outcome, escape, expiry, and requalification navigable as one durable lineage.

Experiments worth running now

The most useful experiments do not ask whether an agent appears generally trustworthy. They search for the boundary where a specific delegation fails.

Experiment	Signal to preserve	Failure that should reopen delegation
Blinded human-agent disagreement review	decision, rationale, evidence accessed, later outcome	confident convergence on the same wrong proxy
Counterfactual and perturbation suite	which changed fact changes the decision	irrelevant wording or identity cue reverses a consequential result
Calibrated abstention trial	confidence band, abstention, escalation, outcome	the agent guesses where the policy requires “unknown”
Diverse red-team panel	distinct model/prompt/evidence lineage and discovered defects	apparent quorum built from correlated copies
Shadow-mode sign-off	what the agent would approve without causing the effect	unacceptable false approvals or systematic human overrule
Delegation canary	action that no valid scope authorizes	the agent or downstream system accepts it
Delayed audit and replay	reconstructed context, authorization, evidence, effect receipt	a later investigator cannot tell human approval from agent delegation
Judge-replacement overlap	old and new judges score the same fixed and fresh cases	unexplained score movement or slice-specific reversal
Longitudinal extension trial	complexity, duplication, coupling, change amplification, later repair cost	local passes coexist with worsening extension cost
Expiry and revocation drill	time from trigger to blocked effect and reopened lineage	stale qualification continues authorizing work

More efficient failure discovery, uncertainty-aware selection of evaluation cases, and anti-reward-hacking training are active research directions rather than solved controls (ProEval, 2026; MONA, 2025). Oversight itself must be tested: frontier evaluations explicitly consider whether an agent can disguise unwanted actions from an automated monitor (Anthropic, 2025).

Open questions the glassbox should preserve

1. How should a project measure semantic error when authoritative outcomes arrive months later or never become fully observable?
2. Which forms of reviewer diversity reduce correlated error rather than merely add tokens and ceremony?

3. How can delegation shrink, expire, and be revoked across composed graphs without breaking reproducibility?
4. When does human review improve a decision, and when does automation bias, fatigue, or deference turn it into a rubber stamp?
5. Which decisions can use outcome-based calibration, and which concern values whose legitimacy cannot be reduced to predictive accuracy?
6. How should a later auditor distinguish an agent-authored recommendation, a delegated machine approval, and a person's own judgment?
7. Can an automated overseer remain useful when the worker is better at the domain, persuasion, and oversight evasion?
8. Which outcome signals reveal evaluator rot early enough to prevent a whole lineage of locally approved but increasingly unmaintainable changes?
9. How should one escaped defect narrow authority and reopen dependent claims without treating every historical approval as equally contaminated?

These are not empty spaces awaiting confident prose. They are design inputs. The glassbox should preserve enough identity, policy, dissent, uncertainty, and outcome evidence for future answers to revise today's graph.

‡ What the reader should keep

1. **Research a bounded delegation, not general trust.** Name the decision class, instrument, evidence, effect, exposure, and expiry.
2. **Test the measuring system.** Judge replacement, contamination, correlated error, and oversight evasion can invalidate an apparent improvement.
3. **Prefer experiments that can revoke authority.** Shadow decisions, perturbations, canaries, delayed audits, and expiry drills expose boundaries.
4. **Preserve unknowns as design inputs.** Abstention, dissent, missing outcomes, and unresolved risk should remain queryable rather than coerced into success.
5. **One escape should have a lineage.** Future controls need to find the claims, decisions, runs, and effects that depended on a failed evaluator.

REFLECTION

What observation would make you narrow the delegation even if its average approval accuracy continued to rise?

Reading the Result Months Later

Claim status: a reconstruction of the CHECKOUT RECOVERY teaching case from shipping provenance mechanisms plus a proposed decision receipt that does not yet ship as a first-class object.

The question

Months after a release, can a person reconstruct what agents did, which contract bit, who exercised authority, what changed, and where the surviving record stops?

The thesis

Auditability is not omniscience. A useful glassbox lets a later reader traverse the declared route from effect back through decision, run, evidence, and work, while making missing conversations, unverified inspection quality, and external truth visible as boundaries rather than filling them with inference.

Reconstructing CHECKOUT RECOVERY

A later investigator begins with the release effect and follows reverse references into the second run-case. The run pins a graph identity, records its route, preserves review carriers, and points to the five-percent release task. The decision receipt is not first-class, so the investigator also reads the workflow transition and comments to reconstruct authority and rationale. The earlier hold remains in history and explains why the later packet contains an operations runbook and rollback threshold.

The reconstruction is useful and incomplete. It can establish who recorded what, which checks and handoffs were present, which route advanced, and how the decision changed. It cannot recover an undocumented conversation, prove every screenshot was inspected carefully, or infer customer safety from the terminal state alone.

That incompleteness is not an embarrassment to hide at the end of the book. It is the measure of the glassbox. The system tells a later reader where knowledge ends.

‡ What the reader should keep

1. **Auditability is not omniscience.** A reconstruction should distinguish stored facts, accountable judgments, external receipts, and missing records.
2. **Read effects backward.** Traverse decision, run, carrier, evidence, and work lineage instead of trusting the terminal state.
3. **Supersession preserves learning.** The later release does not erase the earlier hold or the evidence that changed the governing decision.
4. **A mature glassbox keeps uncertainty visible.** It makes green claims narrow, routes reconstructable, hard failures non-negotiable, and unknowns difficult to erase.

REFLECTION

Which fact in this reconstruction comes from a stored event, which comes from an accountable decision, and which remains unknowable because no durable receipt was ever created?

Future Directions: From Glassbox to Supervised Factory

Claim status: doctrine/design informed by the current product boundary and the research ledgers. The architecture in this chapter is not a claim that Lettuce already ships a factory supervisor, an effect gateway, or delegated semantic authority.

The question

If Lettuce is a durable map, ledger, and refusal surface today, what would it need to become a practical “dark software factory on steroids” without turning autonomy into an excuse to hide uncertainty?

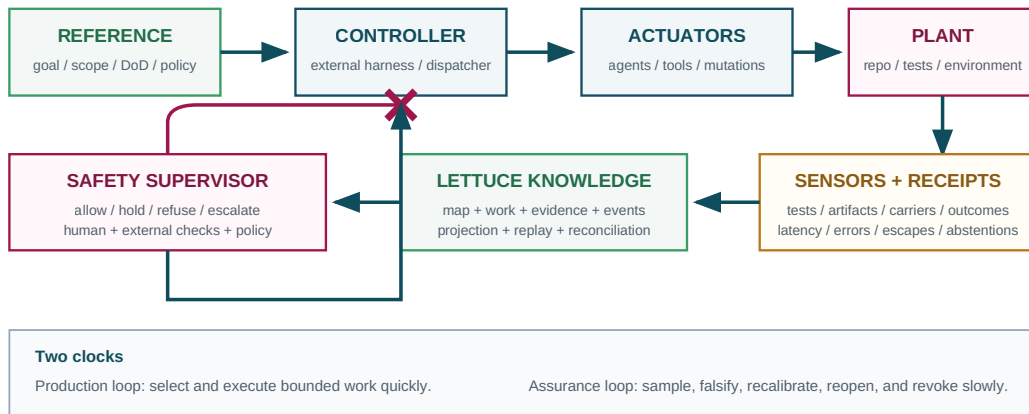
The thesis

The next step is not a more persuasive agent and not a larger task board. It is an evidence-driven control plane around the existing store: bounded dispatch, independent assurance, explicit authority, guarded effects, and a feedback loop that can reopen yesterday’s green claim. A mature factory should be quiet for a human because its routine work is bounded and inspectable, not because the human has been removed from consequential judgment.

This is the natural continuation of the book’s argument. The early chapters ask what a claim means. The middle chapters show how a map, evidence, and graph can make selected transitions durable. The authority and failure chapters show why those mechanisms still need a boundary around them. This chapter asks what that boundary should become if agents are to work continuously without making the system less honest.

The ledger is memory; the factory still needs a supervisor

A controlled shift closes the loop from policy to action to evidence to the next authorized move.



Control-plane reprise. Lettuce is the durable knowledge and provenance boundary. A controller, effect boundary, and assurance loop are additional responsibilities; they must not be smuggled into the current store by implication.

Checkout Recovery, one step further

Return to the running case. Today, an agent can orient to the declared cells, take a lease, exercise a bounded check, attach evidence, meet or fail a gate, and leave a route that another reader can inspect. The difficult question is what happens when this work continues overnight and the world changes while nobody is watching the board.

A supervised factory would not simply ask another agent to “keep going.” It would issue a bounded shift: these cells, this repository revision, this graph definition, this tool and credential budget, this maximum duration, these permitted effects, and this stop policy. The agent would return a shift receipt, including work selected, work refused, evidence produced, unresolved questions, and the reason the controller stopped. A slower assurance pass would then sample the release claim, compare an independent review, and reopen the payment-timeout cells if later evidence showed that the customer-facing explanation was misleading.

Nothing in that description makes the agent wise. It makes the delegation narrow enough to inspect and the failure consequential enough to find its way back into the map.

Start from the kernel, not the slogan

The future is credible only if it preserves the distinctions the current product already makes. Lettuce can give an agent and a human a durable place to look for:

Current kernel can record or refuse	It still does not establish
Scopes, units, dimensions, cells, tasks, leases, and revisions.	That the declared scope includes every important surface.
Evidence links, freshness, depth, gates, DoD projections, and regressions.	That an artifact is truthful or that a human inspected it well.
Graph definitions, carriers, run events, routes, and refused transitions.	That a caller-supplied execution policy is authoritative unless runtime binds it.
Agent-attributed work and human-facing boards, replay, and visual projections.	That an attributed actor was authenticated, authorized, human, or entitled to decide.
Recorded effect assertions and links back to work.	That an external system received, applied, or safely reversed the effect.

This table is not a disclaimer tacked onto the future. It is its design input. Every proposed layer must strengthen one of the left-hand records or close one of the right-hand gaps without changing the meaning of the record already there.

The bounded-shift contract

“Run autonomously” is too vague to authorize anything. A shift should be a versioned object with at least these fields:

1. **Reference:** goal, project revision, map/DoD revision, and graph identity.
2. **Scope:** eligible cells and explicit exclusions, with a denominator that cannot silently expand while the shift is running.
3. **Capability:** agent/model/tool identity, permissions, network boundary, and required skill or policy revision.
4. **Budget:** wall time, token/tool calls, concurrent work, retries, and effect classes.
5. **Stop policy:** refusal, abstention, contradiction, stale evidence, saturation, human decision, timeout, and recovery behavior.
6. **Receipt:** selected and refused work, mutations, evidence, decisions, effects, unresolved questions, final reason, and source revision.



S Contract card: bounded autonomous shift

S SHAPE

A controller grants a finite work packet to an agent and receives a receipt, refusal, or timeout.

C CONTRACT

The packet pins scope, revisions, capability, budgets, permitted effects, and stop policy before work begins.

B BEHAVIOR

Selection, action, evidence, interruption, recovery, and stopping remain attributable and replayable.

X CONSTRAINTS

No silent scope expansion, budget reset, timeout-as-success, or post-expiry mutation.

A AGENT INSTRUCTION

Work only inside the packet; surface unknowns and stop reasons instead of inventing completion.

H HUMAN BOUNDARY

A named authority sets the risk and effect policy; the controller cannot grant itself broader authority.

The receipt matters as much as the dispatch. Without it, a controller is merely an invisible scheduler and a successful-looking run cannot be distinguished from abandoned work, a timeout, or a worker that exceeded its authority.

What “on steroids” should mean

The phrase is useful only if it names measurable improvements. A future system should reduce:

- time for an agent to orient to the current frontier and its constraints;
- unsupported transitions, duplicate work, and conflicting edits;
- time from a discovered defect to a narrowed scope and reopened claim;
- human effort spent reconstructing routine evidence; and
- unsafe or unauthorised external effects.

It should increase the proportion of work that is **bounded, attributable, replayable, interruptible, recoverable, and honest about unknowns**. More agents, more tokens, or a greener dashboard are not success measures by themselves.

A layered target architecture

The layers below are deliberately separable. Lettuce can remain a useful filesystem-native core even when a deployment chooses a different controller or effect gateway.

Layer	Responsibility	Non-negotiable boundary
Reference	Versioned goal, scope, denominator, dimensions, units, DoD, risk, and authority.	A work item cannot silently redefine the quality space it is supposed to satisfy.
Map and frontier	Cells, gaps, stale claims, dependencies, WIP, and next eligible work.	Discoveries can reopen or expand the map; completion is not permanent truth.
Controller	Select bounded shifts, match capability, enforce budgets, and stop on uncertainty or saturation.	The controller is a policy subject, not an oracle; every choice needs a receipt.
Agent boundary	Sandbox tools, credentials, network, repository mutation, and rollback.	Identity does not imply authority, and a proposed effect is not a delivered effect.
Assurance	Independent checks, calibration, expiry, sampling, red-team work, and reopening.	The worker cannot be the sole verifier of its own semantic success.
Effects	Authorize, preview, idempotently execute, and independently receipt external changes.	A Lettuce event records a claim about an effect; it cannot make an external system obey.
Projection	Human and agent boards, graph views, replay, queues, decisions, and anomalies.	Visualization is a projection with source revision, not an additional truth.

This layering also answers a common design mistake: putting every desired feature inside the graph. Graphs should describe and enforce local route contracts. A supervisor governs a shift across graphs; an effect gateway governs the world outside the store; an assurance service tests whether the contract is still worth trusting.

What belongs where

The separation is practical, not merely architectural vocabulary:

Question	Best home	Why
What quality space and DoD are we claiming?	Lettuce reference and map	The denominator and its revisions must travel with the work.
Which eligible work should happen next?	A controller using Lettuce state	Selection needs budgets, capacity, priorities, and stop rules that span tasks.
May this agent read, write, deploy, or notify?	Agent sandbox and effect gateway	

Question	Best home	Why
		A task identity is not a credential and a local event is not a receiver acknowledgement.
Did the check establish its narrow predicate?	Lettuce evidence and graph gates	The result should be durable, attributable, and replayable.
Is the predicate still a good proxy?	Independent assurance and human authority	Adequacy, drift, and value conflicts cannot be delegated to the worker alone.
What can a person see and challenge?	Lettuce projections	Boards, replay, decisions, and anomalies should expose source revision and missing receipts.

Lettuce therefore remains useful without becoming a scheduler, model router, or cloud control plane. A deployment may put those capabilities beside it, but the interfaces between them must be versioned and evidenced. “The controller knew” is not a receipt.

The two-clock factory

The production clock keeps useful work moving: orient, select a cell, acquire exclusive work, act, attach evidence, reconcile, and stop at the declared boundary. The assurance clock is slower and deliberately adversarial: sample green claims, compare independent evaluators, inspect escaped defects, measure correction latency, expire qualifications, and reopen dependent cells.

Conflating the clocks creates two opposite failures. If assurance blocks every small action, people route around it. If production is allowed to define its own assurance, the system rewards throughput and calls the resulting residue quality. The clocks must share lineage but not authority.

The distinction also changes what “done” means. Production can report that a declared predicate passed at revision R. Assurance can later report that the predicate was weak, stale, correlated, or no longer sufficient for the changed scope. A mature system preserves both statements and makes the second one able to reopen the first. It does not edit the old green into a retrospective red.

Map and DoD evolution need the same discipline. A new dimension, unit, or quality bar is a change to the reference layer, not a convenient way to make a failed release appear complete. The change should name its authority, effective revision, affected cells, and migration or re-review rule. Historical results remain interpretable against the old denominator while future work uses the new one.

A maturity sequence, not a promise

The safest roadmap is a sequence of narrower capabilities whose receipts can be tested before the next delegation is granted:

1. **Durable ledger:** keep current filesystem records coherent, versioned, and queryable; make every public claim source-backed.
2. **Trustworthy refusal:** make effective graph contracts authoritative at runtime, require conforming carriers and verdicts, and refuse weakened caller parameters.
3. **Bounded shifts:** add a controller contract for WIP, budgets, capability matching, stop conditions, leases, recovery, and a complete shift receipt.
4. **Independent assurance:** run checks in a distinct trust domain, preserve disagreement and abstention, and make qualification expire or revoke.
5. **Guarded effects:** place deployments, payments, notifications, and other consequential mutations behind authorization, preview, idempotency, independent delivery receipts, and rollback or compensation plans.
6. **Outcome adaptation:** connect later defects and customer outcomes to the claims, decisions, and policies that enabled them, then narrow authority or reopen lineage without rewriting history.

Each stage should ship with a refusal, a recovery exercise, and a replayable receipt. A feature that only adds a happy-path command is not a maturity stage.

The sequence is intentionally conservative. A bounded shift without trustworthy refusal only accelerates bad transitions. Independent assurance without a decision and effect boundary creates reports nobody can act on. Guarded effects without outcome feedback record safer actions but cannot learn whether the policy was adequate. Each stage supplies the missing precondition for the next one.

The hard theoretical limits

Three limits remain even in this architecture. First, a complete map cannot be proven complete from inside the map; scope discovery needs external signals and human challenge. Second, semantic quality is not reducible to a single score: the $-1 \dots 0 \dots +1$ continuum is a useful language for harmful, unknown, and useful states, not a current Lettuce field or universal metric. Third, a model's confidence is not a calibrated probability of truth unless an evaluated, changing population supports that interpretation.

The practical response is not to give up. It is to make uncertainty expensive to erase: require abstention routes, preserve dissent, distinguish assertion from verification, bind evidence to revisions, qualify evaluators on fresh cases, and let an escaped defect find every decision and effect that depended on it.

This is where the $-1 \dots 0 \dots +1$ language earns a precise place. It can help a team say that a cell is harmful, unresolved, or useful while a binary gate says only whether one predicate passed. It must not be smuggled into a single quality number that ranks unlike

claims or converts a human value judgment into a mechanical fact. A future implementation could record such a continuum only if its dimensions, evidence, authority, and revision semantics were explicit.

Long-term maintainability adds another constraint: rot is not only old evidence. It is a changing relationship between the map, the evaluator, the repository, and the world. A future assurance loop should watch for escaped defects, repeated reopenings, rising correction latency, shrinking reviewer diversity, and increasing work that never reaches a meaningful outcome. Those signals do not prove rot, but they should reduce autonomy, trigger fresh evaluation, or force a human decision before the next consequential shift.

What would make the factory trustworthy?

Before claiming a dark factory, ask for these receipts:

- a controller can show why it selected work and why it stopped;
- a worker cannot weaken the contract it is meant to satisfy;
- a second trust domain can disagree and block or narrow authority;
- an external effect has a receiver-side receipt, not only a local assertion;
- a later defect reopens the relevant lineage and changes future policy; and
- a human can reconstruct the route without treating missing records as proof.

These are acceptance questions for a future design, not a checklist that the current binary silently satisfies. The current product already gives the book a useful kernel: filesystem-native records, declared work, leases, evidence, workflow refusal, graph definitions, run histories, carriers, and projections. The proposed controller, trust-domain adapter, receiver-side effect receipt, and outcome-linked reopening remain product work or research.

Until those receipts exist, “dark factory” is an aspiration. The honest product is a glassbox that makes the missing controls visible and gives agents a better place to work while those controls are built.

The design test

The future direction can be reduced to one test. Give a fresh agent a changed Checkout Recovery repository, an incomplete evidence packet, and a bounded authority token. Can it orient, select safe work, refuse the unsupported release, recover after interruption, and leave a record that a human can challenge? Then run the assurance clock after a simulated escaped defect. Can the system find the dependent cells, narrow the authority, and preserve the earlier decision rather than laundering the history?

If not, the missing capability is not “more autonomy.” It is a missing contract, receipt, authority boundary, or recovery path. That diagnosis is the point of a glassbox: it turns an impressive aspiration into a falsifiable next piece of engineering.

‡ What the reader should keep

1. **Make autonomy a bounded control problem.** Define authority, budget, stopping, recovery, and effects before increasing delegation.
2. **Keep production and assurance independent.** Fast work and slow falsification need shared provenance but different incentives.
3. **Treat the ledger as memory, not magic.** A durable claim still needs a meaningful predicate, an honest denominator, and a route to revision.
4. **Measure correction, not activity.** Throughput without reopening, recovery, and outcome learning is a busy factory, not a mature one.

REFLECTION

Which layer would fail first in your current delivery system: the map, the controller, the verifier, the effect boundary, or the assurance clock? What receipt would let you know?

Appendix: A Working Glassbox Vocabulary

These definitions follow the book's use of Lettuce. Each names the nearest idea it must not be confused with.

Artifact. A versioned task-local collection of files and metadata. An artifact makes material inspectable; it does not by itself establish the claim the material is offered to support.

Assertion. A statement supplied by an actor. Attribution and persistence can make an assertion auditable without making it true.

Carrier. A write-once value produced at a graph edge and visit, optionally enriched with typed references. A carrier is the handoff record, not the underlying artifact and not proof of semantic adequacy.

Cell. One coordinate in the coverage model: a unit or other members intersecting an applicable quality dimension within a scope. A cell is an address for a question, not a task and not a universal quality score.

Concern. A graph node's structural role, such as producer, reviewer, human, fork, join, or terminal. Concern determines shape and selected runtime rules; it is not the worker's instruction.

Contract. A declared boundary stating what is required, admitted, refused, or recorded. A contract can be mechanically strong and still target an insufficient proposition.

Decision receipt. The proposed durable record of question, authority, alternatives, evidence, rationale, outcome, conditions, effects, and supersession. Lettuce currently records fragments of this object; a general first-class receipt does not ship.

Definition of Done (DoD). A computed project and per-scope policy over grade, optional depth and recency, milestones, and non-terminal tickets. It is a declared measurement rule, not a semantic certificate.

Depth. The number of distinct store revisions that independently confirm a cell grade under the current evidence rules. Repetition at one revision is not depth.

Dimension. A named quality question and methodology, such as accessibility, resilience, or agent experience. A dimension guides what to examine; it is not proof the examination happened.

Effect. A run advance's declared reference to an object the step changed. Current effects support forward and reverse provenance but are executor assertions, not automatically mutation-issued receipts.

Evidence. Material or references offered in support of a claim. Evidence strength depends on provenance, relevance, freshness, independence, and the question it is used to answer.

Execution policy. The declared model/capability token, effort, and context residency for an agent graph station. It records how a station is intended to run; it does not prove compliance or capability.

Freshness. A read-time category derived from store-revision distance since evidence assertion. It detects movement around a claim; it is not wall-clock age or automatic semantic invalidation.

Gate. A mechanical prerequisite that can refuse a transition or handoff. A gate is useful when its predicate is narrow and consequential. It is not a person, review method, or general quality judgment.

Graph definition. A versioned, soundness-checked coordination shape with node behaviors, policies, edges, guards, loops, and reusable subgraphs. It declares allowed coordination; it is not an execution history.

Grid. The declared cross-product of units and applicable dimensions for a scope. It provides an honest denominator by counting implicit untested cells; it does not guarantee the chosen surface is complete.

Instruction. A graph station's behavioral contract in words. It says what a worker should do; it is guidance unless paired with evidence and enforceable postconditions.

Lease. An explicit, expiring claim to exclusive task work. It coordinates actors; it does not imply progress or ownership of the underlying product area.

Methodology. Guidance describing how to examine a dimension. It can improve agent behavior and review consistency but is not an executable receipt.

Provenance. The attributable relationship among an action, actor, input, output, version, and route. Provenance makes later questions answerable; it does not establish semantic truth.

Refusal. A failed declared predicate that leaves the protected state unchanged. Refusal demonstrates that one boundary bit; it does not show the full process is sufficient.

Run-case. One event-sourced enactment associated with a graph. Its state is derived from events. A run-case is not a task, an agent process, or a guarantee that external effects occurred as described.

Scope. A named boundary over which units, applicable dimensions, grid, and DoD policy are interpreted. A scope partitions a quality claim; it is not merely a label or team name.

Supersession. An explicit relation in which a later decision or version governs after an earlier one without erasing history. Supersession differs from editing the old record in place.

Task. An evented work item with workflow, comments, leases, runs, artifacts, and relationships. A task organizes accountable work; completing it does not automatically harden a cell or satisfy the project DoD.

Unit. A thing within a scope to which applicable dimensions are applied, such as a command, screen, service, or workflow. A unit is part of the coverage denominator, not necessarily the same size as a task.

Workflow. A named set of task states and validated actions. Workflow controls task lifecycle; graph definitions coordinate a potentially richer enactment across stations and handoffs.

Appendix: Verified Product Receipts

This appendix preserves compact checks behind selected shipping claims. It is not an operating tutorial. Agents obtain current syntax from `lettuce skill`, `lettuce usage`, and the embedded wiki. A reader uses these receipts to ask a different question: what observable result justified the prose?

Every receipt has five fields:

- **Cutoff** names the product revision and built binary.
- **Subject** names the narrow property under examination.
- **Stimulus** describes the fixture or comparison without teaching command choreography.
- **Observed result** states what actually happened.
- **Boundary** states the larger conclusion the result cannot support.

The common cutoff for this edition is product `main` at `165de048e`, exercised on 2026-08-10 with binary `v0.15.0-4126-g616fe2f8b-dirty`. The suffix includes this documentation branch; product code was rebased over the named `main` cutoff.

Receipt R1: the book graph is sound and clean

Field	Receipt
Subject	The <u>CHECKOUT RECOVERY</u> teaching graph can be stored as a sound definition and carries evidence into its review station.
Stimulus	A fresh isolated store accepted <code>checkout-recovery-review.json</code> ; the graph linter then inspected the stored definition.
Observed result	Creation succeeded with effective hash <code>a36a998f...805e</code> ; lint returned <code>clean: true</code> with no smells. The generated Mermaid source matches the checked-in node order and concern shapes.
Boundary	Sound and clean do not mean the review instruction is wise. The current diagram also omits the declared <code>timeout-evidence</code> carrier, so the stored spec remains part of inspection.

Receipt R2: an empty denominator is not zero percent

Field	Receipt
Subject	A rollup with no applicable, non-excluded cells reports no ratio rather than fabricating failure.
Stimulus	Focused operations tests exercised an all-excluded member, the blended headline, and a non-empty control population.
Observed result	The empty cases returned a nil ratio rendered as <code>n/a</code> ; the populated control retained its real numeric ratio. Tests <code>TestLET1409EmptyDenominatorRatioIsNotApplicable</code> , <code>TestLET1409BlendedHeadlineDoesNotFabricate</code> , and <code>TestLET1409HeadlineStillReportsARealRatio</code> passed.
Boundary	<code>n/a</code> does not mean done. It means the selected population contains nothing applicable to count.

Receipt R3: removing evidence stops projection credit

Field	Receipt
Subject	A cell promoted through an evidence-backed gate must not remain credited after its last evidence disappears.
Stimulus	Focused operations tests removed the final evidence, inspected rollup and board behavior, reconciled the cell, and compared a direct-set no-evidence control.
Observed result	Projections stopped crediting the evidence-backed hardened state; reconciliation demoted the stored assertion. The direct-set control remained unchanged. The LET-633 test family passed.
Boundary	This is not a universal proof that every hardened cell has evidence. Facilitated direct assertions form an explicit, separately visible population.

Receipt R4: verification depth resists evidence reuse

Field	Receipt
Subject	Strong verification depth for one cell requires a distinct evidence reference, not merely another call.
Stimulus	Focused depth tests repeated confirmations across revisions, reused evidence on the same cell, and exercised evidence-backed controls.
Observed result	A distinct revision could add ordinary affirmation; reusing the same per-cell evidence did not add another <code>depth_verify</code> unit. <code>TestFresh2DepthDoesNotResurrectAfterRegressRearden</code> and <code>TestDeriveDepthDedupsVerifyByEvidence</code> passed with their sibling depth tests.
Boundary	Distinct references do not establish independent reviewers, distinct methods, or uncorrelated error.

Receipt R5: visual projection is not the full graph contract

Field	Receipt
Subject	Generated graph visualization is deterministic but partial.
Stimulus	The same clean stored graph was rendered to Mermaid after adding the <code>timeout-evidence</code> carrier that removed its lint smell.
Observed result	The diagram reproduced stable nodes, shapes, instructions, and edge order, but the carrier name did not appear.
Boundary	A picture supports orientation. The stored definition, structured show result, lint receipt, and effective hash remain necessary for contract inspection.

How to challenge a receipt

A receipt ages. Rebuild the current binary, repeat the named focused test or fixture, and compare the observed property rather than merely expecting the old text to remain true. When the result changes, update the claim ledger and every chapter that depends on it in the same editorial pass.

Appendix: Claim and Status Ledger

This ledger maps the book’s load-bearing claims to their present evidence and status. It is intentionally selective: a future edition should generate a more granular inventory, but no reader should have to infer whether the central argument describes shipping behavior, design doctrine, or a research proposal.

ID	Claim	Status	Current basis	Recheck trigger
C01	Declared grids keep implicit coordinates in a coverage denominator.	Shipping	Public cell/grid contract; Chapter 4 model.	Grid, pack, or rollup semantics change.
C02	An empty applicable population reports <i>n/a</i> , while a non-empty population may report 0%.	Shipping	Receipt R2; LET-1409 tests.	Ratio or board projection changes.
C03	Freshness derives from store-revision distance and is a relevance proxy, not a clock or oracle.	Shipping + interpretation	Public cell freshness contract; Chapter 5 boundary.	Freshness buckets or revision anchoring change.
C04	Strong verification depth needs a distinct evidence reference per cell.	Shipping	Receipt R4; public skill and focused tests.	Confirmation/depth policy changes.
C05	Removing the last evidence from a gate-promoted cell removes projection credit; reconciliation repairs stored state.	Shipping	Receipt R3; LET-633.	Evidence removal, rollup, or reconcile changes.
C06	Methodology directs attention but does not prove attention occurred.	Doctrine/design	Chapter 7 argument; no product oracle claimed.	A trace or evidence-contract feature ships.
C07	Graph definitions are versioned, soundness-checked contracts with deterministic compiled identity.	Shipping	Public graph contract; Receipt R1.	Definition, compile, or hash semantics change.
C08		Shipping, bounded		

ID	Claim	Status	Current basis	Recheck trigger
	A run-case records one evented enactment and can refuse supplied carrier, quorum, routing, or loop conditions.		Public run-case contract and focused product tests.	Definition-bound enactment closes the caller-supplied seam.
C09	Execution policy changes authored identity but current run-cases do not enforce staffing.	Shipping limitation	Controlled upstream sweep through 165de048e; Chapters 8 and 11.	Runtime begins consuming model, effort, or residency.
C10	A carrier records producer-side availability, not consumer receipt or understanding.	Shipping + doctrine	Carrier ledger behavior; Chapter 9 boundary.	Consumer-issued acknowledgments ship.
C11	Replay refolds stored events; it does not reenact model calls or external effects.	Shipping boundary	Public run/show/viz behavior; Chapter 9.	External execution receipts or reenactment ship.
C12	Several model calls may be correlated and a quorum is not a correctness probability.	Research-backed doctrine	Chapter 12 and probability research ledger.	Outcome-calibrated independence mechanism ships.
C13	A human-shaped node does not establish identity, authority, rationale, or approval.	Shipping limitation	Graph concern contract; Chapter 13.	Structured human authority contract ships.
C14	Lettuce preserves decision fragments but no first-class complete decision receipt.	Shipping limitation + design proposal	Chapter 13; owner feedback Priority 3.	Decision object becomes public behavior.
C15	The -1 ... 0 ... +1 continuum is a proposed inquiry model, not a cell state or universal quality score.	Doctrine/research	Chapter 14.	A calibrated graded-evidence feature is accepted.
C16	Agent semantic authority should be bounded, qualified, expiring, externally authorized, and outcome-linked.	Research-backed doctrine	Delegated Semantic Authority chapter and research ledger.	First-class delegation/qualification ships or evidence falsifies controls.

ID	Claim	Status	Current basis	Recheck trigger
C17	A later audit can reconstruct recorded lineage but cannot recover undocumented judgment or prove external truth.	Shipping boundary	Final reconstruction case; Chapters 9 and 17.	New receipts materially extend the lineage.

Status discipline

Shipping means the named narrow behavior was observable at the edition's cutoff. It does not promise every adjacent path is defect-free. **Doctrine/design** means the book recommends a way to compose mechanisms and judgment. **Research** means the proposition requires evaluation, calibration, or further product work. A mixed label is not ambiguity: it identifies a shipping mechanism and the interpretation the book deliberately places around it.

Software work needs a glassbox.

When people and language-model agents work together, a green task, passing test, or polished graph can create more confidence than the evidence supports.

Inside Huru's Glassbox is a human field guide to the coordination system agents use underneath that work: scopes and coverage maps, evidence and Definition of Done, graph contracts and carriers, replayable decisions, probabilistic workers, and the judgment automation cannot honestly replace.

It does not promise a lights-out factory. It asks a more useful question: what can a system enforce, what can it only record, and where must a person still decide?

