The Epistemology of Repair

Rethinking system failure as a high-value, information-producing event.

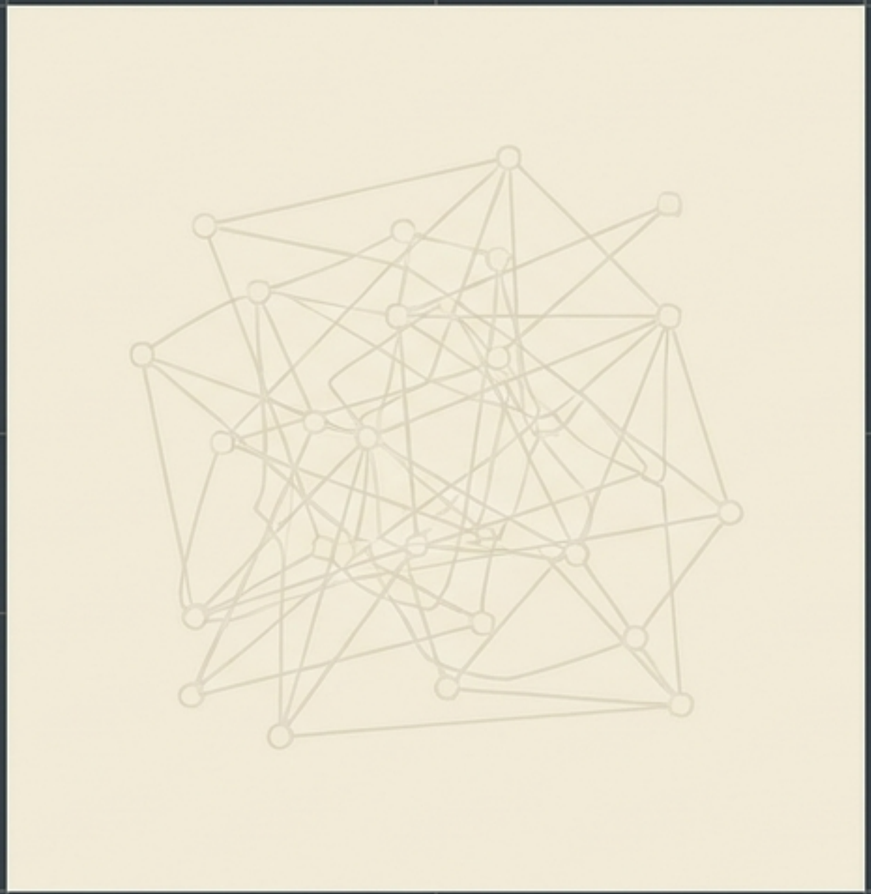
Based on the research of Flyxion (2026).

The Problem with Working Systems

A working system withholds its own internal structure.

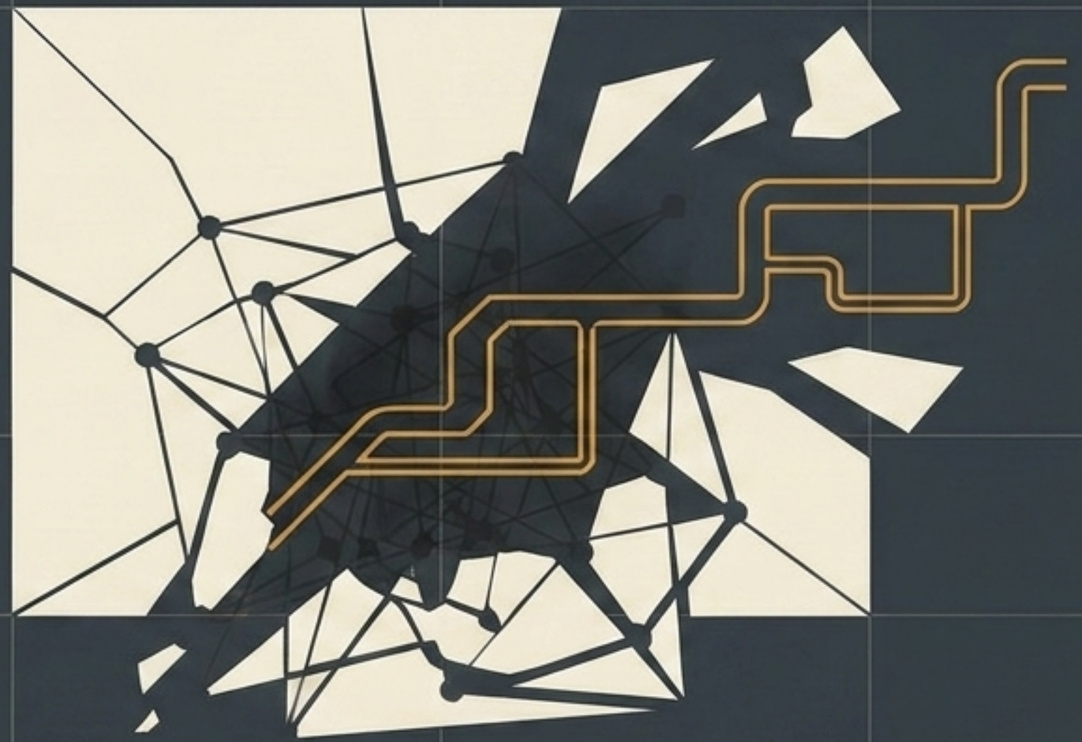
Uptime is compatible with a surprisingly large class of underlying structures, as long as they happen to produce the correct external behavior.

Attention follows failure, not function. The knowledge required to build a system and the knowledge required to explain a working instance are not the same—and the gap between them can persist indefinitely as long as nothing breaks.



Breakage is an Epistemic Event

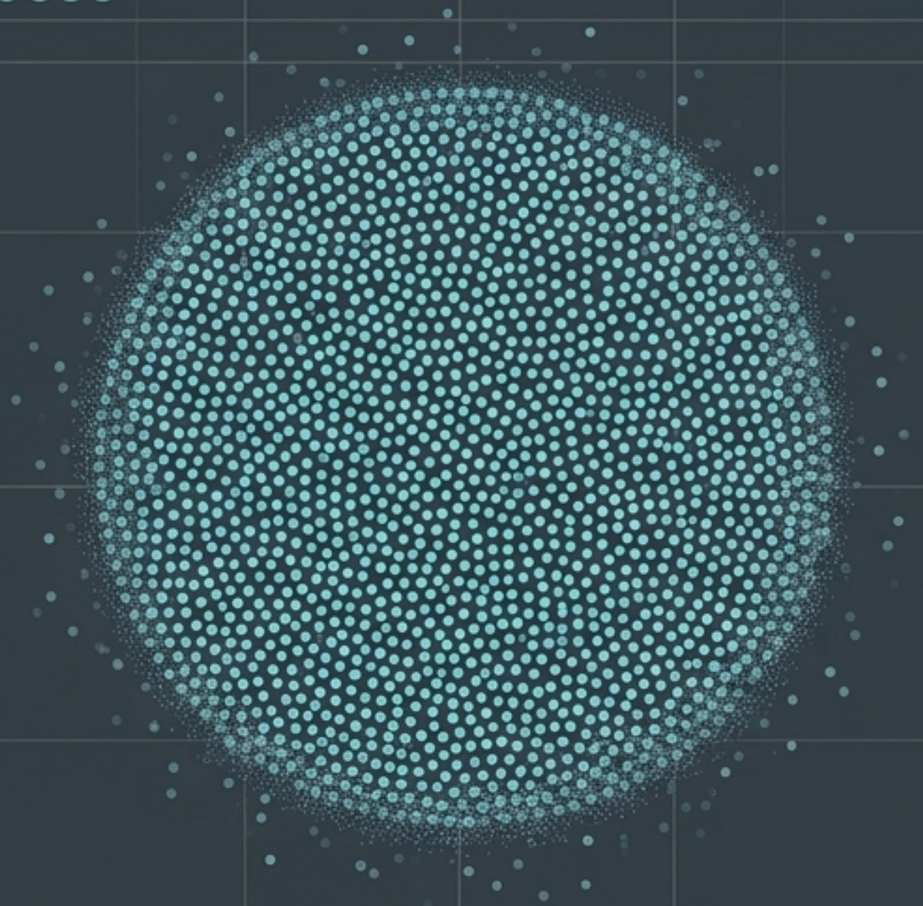
The moment a system stops behaving as expected, the space of compatible arrangements contracts sharply.



Failure rarely damages a system's structure.
Instead, it reveals a hidden constraint that
was always there, but simply had no occasion to matter before.

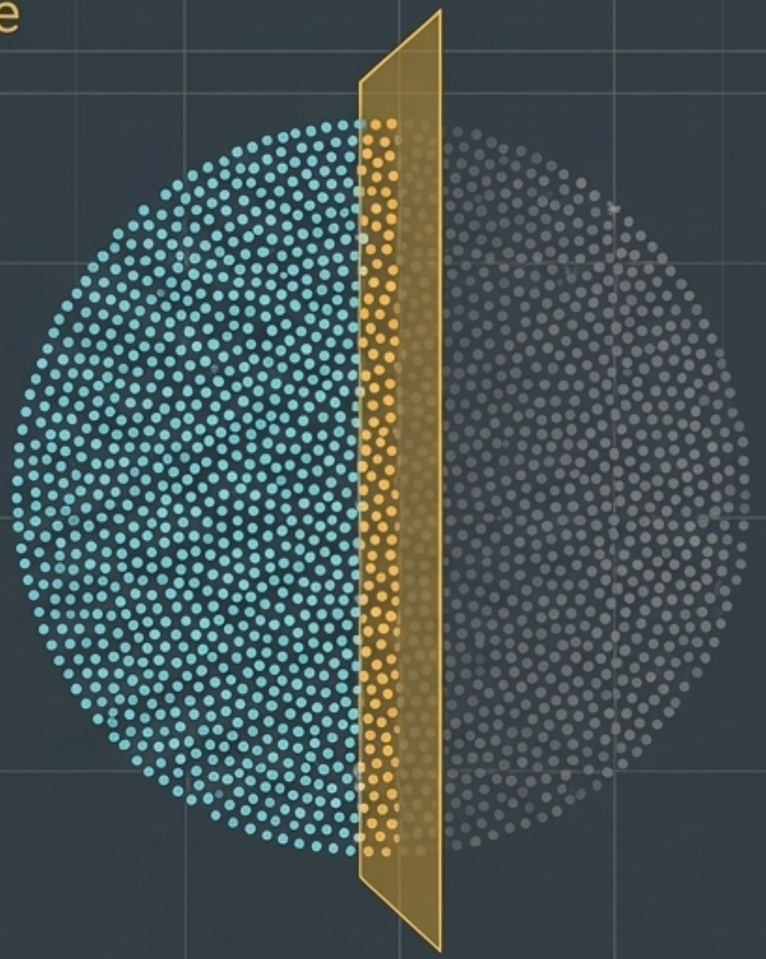
The Evidential Weight of Success vs. Failure

Success



Accumulates weak evidence gradually. A long run of correct executions slowly and unevenly narrows the space of possible structures.

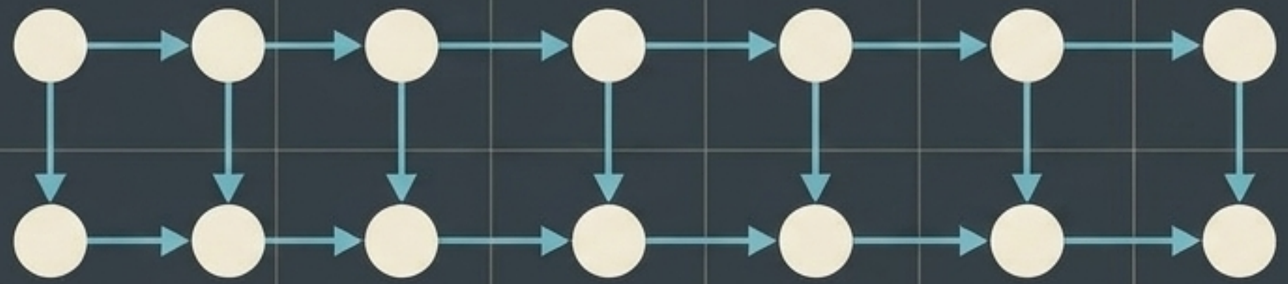
Failure



Produces strong evidence instantly. A single well-characterized failure forces the hypothesis space to be compatible with the specific way the system went wrong.

Inheriting the Experiment

Formal Science



Repair is experimental inference performed on phenomena the repairer did not get to design.

Feyerabend's Opportunism

A scientist designs an experiment to discriminate between live hypotheses.

A repairer inherits an experiment reality has already run—the corrupted heap, the failed build—and must reconstruct the conditions just to interpret what it showed.

The Underlying Procedure in Three Registers

Dependencies	Memory Corruption	Race Conditions
Transitive layers obscure the broken package.	Crash site is arbitrarily separated from the initial invalid write.	The act of observing the symptom alters the timing that creates it.
Incremental version pinning.	Memory sanitizers.	Invariant reasoning & controlled scheduling.
Isolating a hypothesis (reverting A alone restores the property) rather than isolating a component.	Collapsing the temporal gap between corruption and detection.	Causal indistinguishability; the diagnostic tool acts as a perturbation.

Two Competing Objectives

Restorative Value

Does this intervention restore the desired behavior?

High-restorative interventions act broadly, averaging away the exact distinctions diagnosis requires.

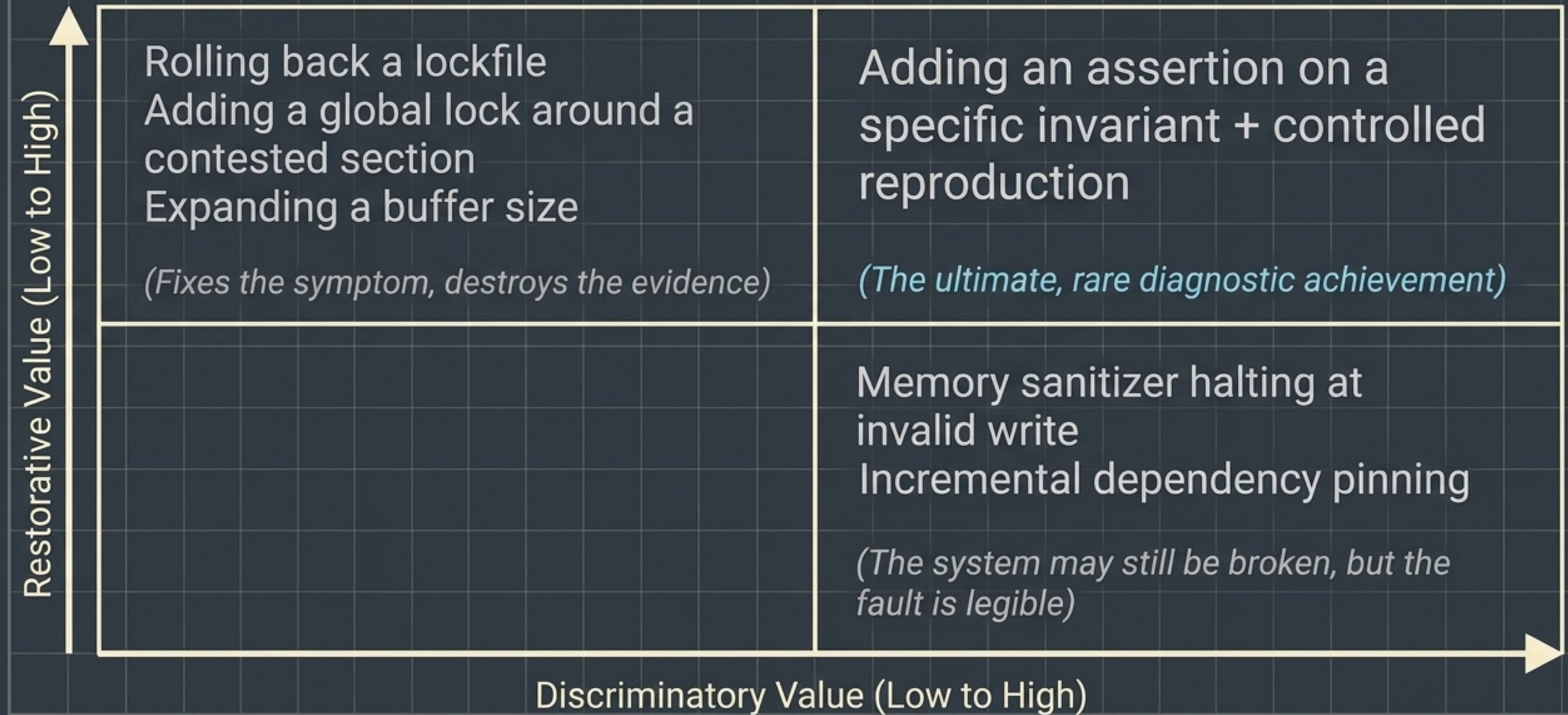
Diagnostic Skill Lives Here.

Discriminatory Value

How much does this narrow the space of competing explanations?

High-discriminatory interventions act narrowly, accepting a slower path to working order in exchange for a legible record of what happened.

Scoring Interventions



The Workaround as a Causal Theorem

A workaround that reliably restores a property—even without a known root cause—is not an absence of knowledge. It is a **verifiable interventional fact.**

Two Paths Diagram

Path A: Rigorous Discovery

One variable isolated, one hypothesis eliminated.
Yields: Falsifiable theorem with high evidential weight.

Path B: Sloppy Discovery

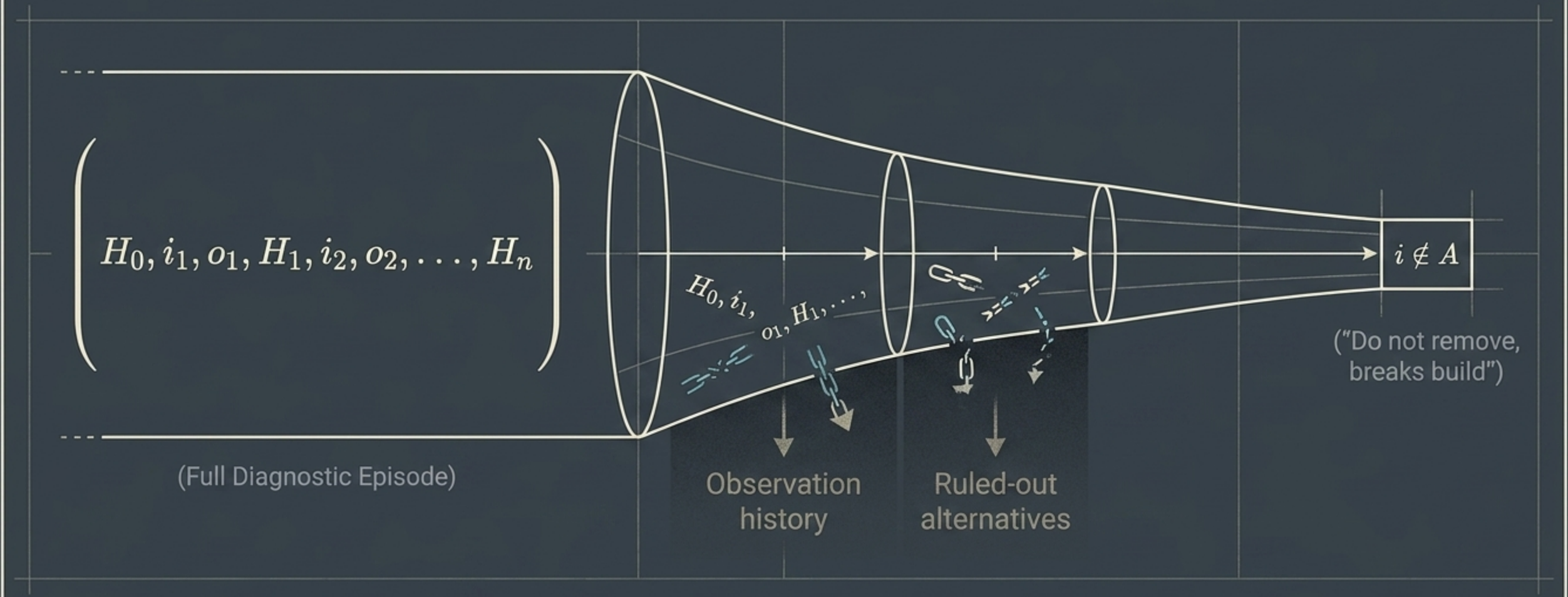
Trying multiple things until the symptom vanishes.
Yields: Function restored, but highly vulnerable to coincidence.

"Mechanism is not the minimum unit of useful causal knowledge. A single well-confirmed intervention is." — Judea Pearl

Accumulator Substrates: Where Information Survives

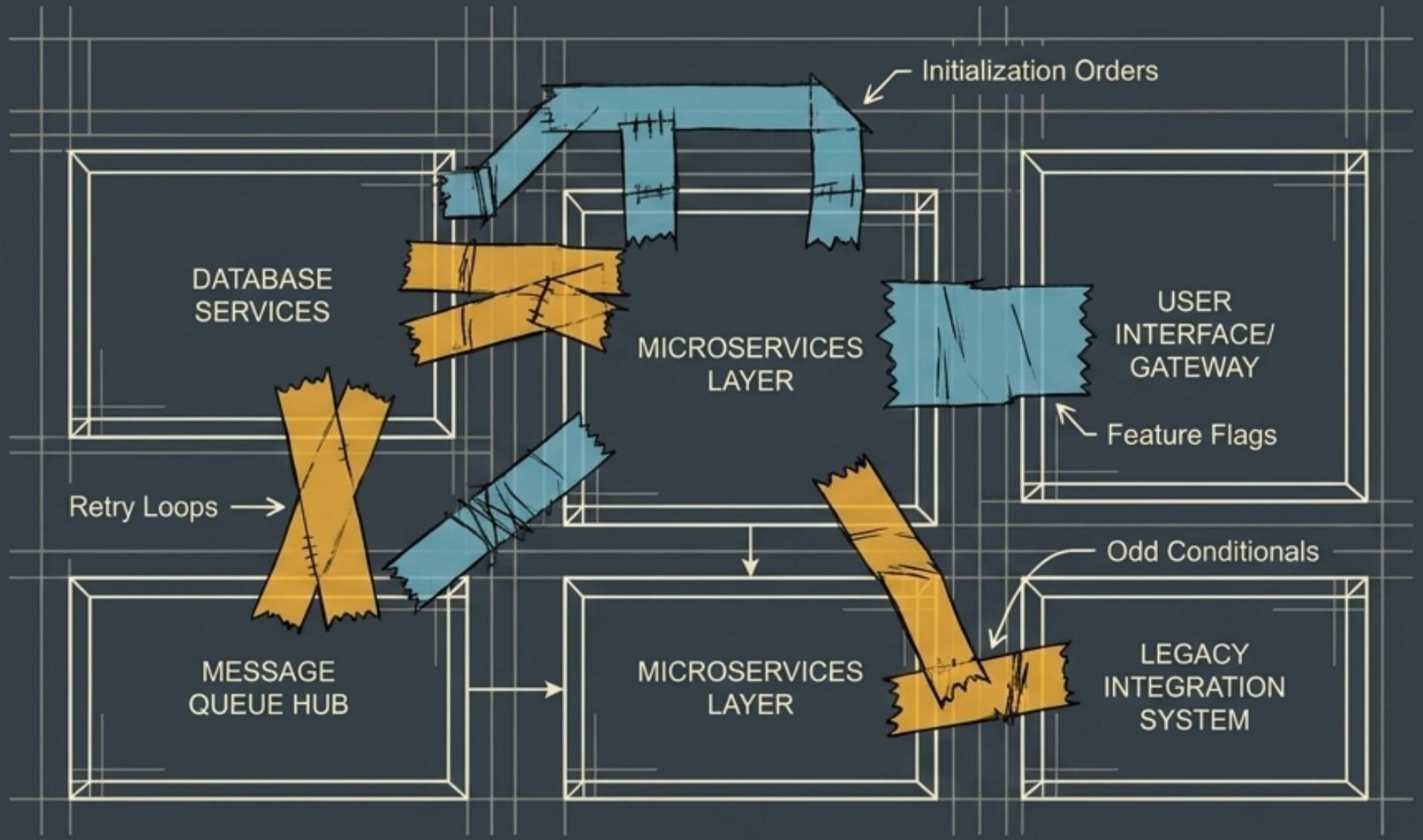
Substrate	Retention Fidelity	Diagnostic Bandwidth	Cost to Write	Cost to Interpret
Terse Commits (‘fix build’)	Low	Low	Low	High
Code Comments (‘do not remove’)	Low	1-bit	Low	High
Postmortems (Preserves ruled-out hypotheses to prevent duplicated effort)	High	High	High	Low
Engineer Intuition	Weighted Avg	Unverifiable	Zero	High Risk

Epistemic Compression



A complete diagnostic episode narrows hypothesis spaces through successive interventions. Over time, accumulator substrates compress this rich history down to a single admissibility judgment, discarding the justification.

The Patchwork of Local Rules



Drawing on Mark Wilson's concept of "physics avoidance", a mature codebase survives as a patchwork of locally valid rules.

Practitioners routinely get correct results by stitching these rules together via convention, successfully avoiding the underlying explanations indefinitely, as the localized constraints keep holding.

Redefining Technical Debt

Structural Debt

- **Definition:** Bad code, slow performance, quick fixes that cost computational resources later.
- **Nature:** Physically limits the system.
- **Payment:** Rewriting code.

Epistemic Debt

- **Definition:** Compressed logic, lost justifications, functional workarounds with forgotten origins (“feared code”).
- **Nature:** Intellectually limits the maintainer.
- **Payment:** Rediscovering the causal justification, not rewriting code.

The Fossil Record of Failure

A mature software system is partly composed of compressed failed experiments. It is a fossilized record of everything that was once allowed to go wrong.

The opposite of ignorance in a complex system is not explanation. It is successful discrimination—the accumulated ability to tell apart states that used to look identical.