

Reasoning System Antifragility Audit – Report

Filled from the generic template against DrawWise's own causal-reasoning subsystem. Every finding below traces to a specific file:line, git commit, or test name — re-check any of them directly rather than trusting this summary; that's the whole point of the methodology this report follows.

Public sample copy: two items (a personal email address and a citation to an internal, not-for-distribution working file) have been redacted from this version. Nothing else was changed — every score, finding, and line-number citation is unedited from the original.

Cover

FIELD	VALUE
System under evaluation	DrawWise (drawwise), causal-reasoning subsystem — repo fourth-pool
Version / commit evaluated	c2ca0b1 (HEAD at audit time), plus one unrelated uncommitted file (server/application-cycle.mjs, out of scope — see §2)
Submission date	2026-08-13 (self-audit, read-only repo access)
Evaluator(s)	Claude (Sonnet 5), direct primary-source read of every file cited below; one Explore subagent used for first-pass discovery, independently re-verified by the evaluator before any finding was written down
Adversarial second reviewer	None — this pass had no second, independent evaluator. Treat every finding as single-reviewer, not adversarially confirmed.
Report date	2026-08-13
Evaluation Charter reference	Defined ad hoc for this run — see §2

1. Executive Verdict

Classification: ROBUST (weighted score 3.45/5.0 — right at this methodology's own Robust / Antifragile-leaning boundary; see the judgment call at the end of §6).

DrawWise's causal-reasoning core is unusually well-built for a product this size: an explicit, versioned, 14-node/18-edge causal graph (causal-graph.mjs) is genuinely injected into the live system prompt (not just documented), backed by a real Pearl-rung-2 intervention layer (intervention.mjs) across three different draw mechanics, a risk bowtie that explicitly refuses to invent a composite score (risk-network.mjs), and — the standout mechanism — an evidentiary-state machine (edge-trust.mjs) that couples a causal edge's real-world corroboration/contradiction history directly to what the system is *allowed to do automatically*, with a real, already-fired production example of exactly the failure mode this whole audit framework exists to catch (a confounded correlation, correctly quarantined rather than either blindly trusted or blindly dismissed — see §5.3).

It falls short of an unqualified **Antifragile** verdict for reasons that are all real and all fixable, not hypothetical: the mechanism doing the most safety-critical work (`edge-trust.mjs`'s authority coupling) is wired to exactly **one** action class in production (an auto-email suppression), its human-closure path (`confirmConfound()`/`restore()`) has no operational entry point anywhere in the running system, and the model-serving/fallback layer that keeps the advisor answering at all under latency stress (`fallback.mjs`) is real, thoughtfully incident-driven engineering with **zero automated test coverage** on five of its six exported functions. Every "self-correction" event this audit found was a human noticing a real incident and patching code well — genuinely good engineering, but not yet the system improving its own model automatically, which is what this methodology's Antifragile band requires.

Confidence: Medium-High. Every claim below is grounded in code I read directly (line numbers cited) or a test I ran myself (`npm test`, verified firsthand, not taken from the subagent's report). The real limitation: I did not perform live stress injection against the running production service (`drawwise.ai`) — that would affect real paying customers without separate authorization — and I did not run the costed behavioral eval harness (`scripts/eval-run.mjs`, real tokens against a real model). Where evidence stops at "the code is built to handle this" rather than "I watched it handle this live," I've said so explicitly.

2. Scope

- **In scope:** the causal-reasoning subsystem specifically — `causal-graph.mjs`, `risk-network.mjs`, `evidence-ladder.mjs`, `intervention.mjs`, `driver-diagnosis.mjs`, `edge-trust.mjs`, `decision-reconciliation.mjs`, `structural-consensus.mjs`, `data-driven-check.mjs`, `observe-update.mjs`, `cross-species-pressure.mjs`, `density-dependence-context.mjs`, `quota-anomaly.mjs`, `scoring-plausibility.mjs`, `cohort-pipeline.mjs`, `consensus.mjs`, `reconcile.mjs`; plus `fallback.mjs` and the relevant `server.mjs` wiring, since they're the layer that determines whether the advisor answers at all under stress.
- **Out of scope:** the ~40 state-specific data/scoring pipeline files (`arizona.mjs`, `colorado.mjs`, `score-elk.mjs`, etc.) except as consumers of the reasoning engine; live stress injection against the production service; the costed live-model eval harness; `server/application-cycle.mjs`, which had uncommitted local changes at audit time (`git status --short` showed it modified, not committed) — scoring mid-edit work would misrepresent unstable work as a finished claim, a straightforward violation of this methodology's own primary-source gate.
- **Access granted:** full read access to the repo, `git log`/`git diff`, and permission to run the local test suite (`npm test`, read-only against the codebase, no network calls, no production side effects).
- **Charter deviations:** this charter was written by the same evaluator running the audit, for a self-audit context, rather than negotiated with a separate client beforehand. Flagged per this methodology's own defensibility standard — a charter authored post hoc by the auditor is weaker evidence than one signed in advance by an independent party, even when the scoping itself is sound.

3. Score Card

DIMENSION	WEIGHT	SCORE (1-5)	WEIGHTED	EVIDENCE REF
Brittleness surface (Stage 1)	15%	4	0.60	\$4.1
Causal structure (Stage 2)	15%	4	0.60	\$4.2
Risk coverage (Stage 3)	15%	4	0.60	\$4.3
Reasoning depth & calibration (Stage 4)	15%	3	0.45	\$4.4
Stress response (Stage 5)	25%	3	0.75	\$5
Guardrail & agency (Stage 6)	15%	3	0.45	\$4.5
Total	100%	—	3.45	—
Deployment & Operational Integrity (Stage 7 — proposed)	not weighted	not assessed	—	\$9

4. Stage Findings

4.1 Stage 1 – Component Inventory

- **Claim:** the causal-reasoning core is explicit, typed, inspectable code, not an LLM improvising causal-sounding prose.
- **Test performed:** read `causal-graph.mjs` in full; grepped `server.mjs` for actual usage of its exports.
- **Evidence:** `causal-graph.mjs:1-19`'s own header states the design intent directly — *"the model never computes a causal answer itself, it reads a small structured graph and cites it"* — and this is genuinely wired live: `graphSummary()`, `preconditionSummary()`, `riskPathSummary()` are imported at `server.mjs:42` and rendered directly into the system prompt at `server.mjs:1931`, `1946`, `1956`. `evidence-ladder.mjs`'s `describeLevel()` is likewise live at `server.mjs:1385`.
- **Falsification condition:** would have been refuted by finding these exports defined but never imported into the live prompt-assembly code — the opposite of what I found.
- **Verdict: confirmed**, with one real deduction. `decision-reconciliation.mjs` and `structural-consensus.mjs` are built and unit-tested standalone but **not imported anywhere in `server.mjs`**, **`consensus.mjs`**, or **`reconcile.mjs`** (`grep -rn` for their exports across those files returns nothing) — live scoring traffic still runs through the older, less-sharp `consensus.mjs` / `reconcile.mjs` pair (confirmed live via `score-elk.mjs:33,36`, `score-whitetail.mjs:20`, `score-muledeer.mjs:42`, `score.mjs:12`). This is honestly disclosed in the code's own commit history (`b49ecf0`: *"built and tested standalone first"*) rather than hidden, which matters for how it's weighted — see §7.

4.2 Stage 2 – Causal Structure Audit

- **Claim:** DrawWise maintains one real, versioned, small causal graph rather than regenerating causal-sounding language per turn.
- **Test performed:** read `causal-graph.mjs` in full (14 nodes, 18 edges); read all 22 test names in `causal-graph.test.mjs`.

- **Evidence:** every edge carries a stated confidence on an explicit 1–3 scale with documented criteria (`causal-graph.mjs:42–50`) and a real mechanism string — enforced by its own test, *"every edge has a real mechanism string, never a bare claim with no explanation"* (`causal-graph.test.mjs:22`). Weight-2 edges are explicitly downgraded from weight-3 when the underlying claim is field-expertise-not-yet-independently-verified rather than published research (e.g. `causal-graph.mjs:62`) — a real, checkable honesty discipline, not decoration. `preconditionsFor()` backward-chains correctly and de-dupes shared ancestors (tested: `causal-graph.test.mjs:112`, the `HARVEST_REGIME` shared-ancestor case). The graph's own size discipline is itself tested (`causal-graph.test.mjs:76`, *"the graph stays genuinely small"*).
- **Falsification condition:** would be refuted by a sampled live "why" answer that doesn't trace to any edge in this file.
- **Verdict:** confirmed for the data structure itself (this alone would earn a 5), but I did not sample live model outputs to confirm the last mile — that every "why" the advisor actually says in production traces to a specific edge, rather than the model occasionally free-associating around the injected summary. That test exists (`scripts/eval-run.mjs`) but costs real tokens and requires a running server; out of scope for this pass (see §2). Scored 4, not 5, specifically because that last-mile claim is unverified, not because the graph itself has a flaw.

4.3 Stage 3 – Risk & Failure-Mode Mapping

- **Claim:** risk modeling is real and explicitly refuses false precision.
- **Test performed:** read `risk-network.mjs` in full; read all 15 test names in `risk-network.test.mjs`; independently searched for failure modes the codebase's own comments/docs don't name as risks.
- **Evidence:** `risk-network.mjs:14–22`'s header explains, with a named alternative it deliberately rejected, why it does *not* collapse heterogeneous threats (CWD %, drought DSCI, snowpack % of median) into one composite score — *"inventing a common denominator... would be exactly the false-precision this product's own design refuses to ship."* This is enforced by test: `risk-network.test.mjs:139`, *"riskNetworkFor: never invents a composite risk-exposure score."* `cross-species-pressure.mjs`, `quota-anomaly.mjs`, `density-dependence-context.mjs`, `scoring-plausibility.mjs` all return `null` rather than fabricate when data is insufficient, and each has a real test confirming the honest-null path (e.g. *"densityDependenceContextFor returns null for antelope — the ND source population is mule deer only"*).
- **Falsification condition:** would be refuted by finding a plausible failure mode with no corresponding detector anywhere in this list.
- **Verdict:** confirmed, with two real gaps this audit itself surfaced (not previously named as risks anywhere I found in the codebase's own docs/comments): (1) `fallback.mjs`'s failure-classification logic is untested (§4.5/§5.2); (2) the two generalized-but-unwired mechanisms from §4.1 mean the sharper materiality/structural-mismatch protections aren't actually protecting any live decision yet. Per this methodology's own Stage 3 definition, *the gap between the independently-found list and the system's own register is the finding* — these two gaps are that finding.

4.4 Stage 4 – Reasoning-Depth Test

- **Claim:** the system reasons at Pearl's intervention rung (not just association), with calibration-aware confidence.
- **Test performed:** read `intervention.mjs` in full (all three draw mechanics + `decisionCardFor()`); read `evidence-ladder.mjs`; read `scripts/eval-run.mjs`.
- **Evidence:** `bankingOutlookFor()` / `mtBankingOutlookFor()` / `coBankingOutlookFor()` each answer a genuine rung-2 question ("what happens if I bank a point instead of applying now") with real published-table lookups, never a forecast, and each carries a distinct caveat proportional to real data quality — e.g. Colorado's `creep_reliable` gate (`intervention.mjs:246`) explicitly refuses to trust a smoothed rate when year-to-year swings are too large, and the Wyoming `ABANDON` action (`intervention.mjs:441`) requires *both* a climbing cutoff *and* low volatility, deliberately stricter than either alone — a real, documented anti-overconfidence

design (`intervention.mjs:430-441`). `evidence-ladder.mjs` itself states plainly that only Level-1 evidence is real today, refusing to claim higher-level verification the product doesn't have (`evidence-ladder.mjs:8-14`). A real behavioral eval harness exists (`scripts/eval-run.mjs`) with a case built specifically to catch confabulation — `"no-herd-data-is-honest-not-fabricated"` — but it holds only 2 cases and requires real spend against a live model; I did not run it.

- **Falsification condition:** would be refuted by finding the advisor's stated confidence uncorrelated with its actual hit rate over time.
- **Verdict: plausible, not confirmed at 4/5.** Rung-2 intervention logic is genuinely strong and well-calibrated *by construction* (thresholds are real, stated, and conservative). But I found no mechanism anywhere that tracks whether a "high confidence" call is actually right more often than a "low confidence" one over time — confidence labels are rule-derived from data-quality thresholds, never checked against realized outcomes. That's the specific gap the calibration half of this dimension's rubric anchor requires and I could not find evidence for it either way. Scored 3.

4.5 Stage 6 – Guardrail & Agency Audit

- **Claim:** autonomy narrows automatically when evidence quality drops, and a human can exercise real override.
- **Test performed:** read `edge-trust.mjs` in full; grepped every call site of `authorityFor`, `confirmConfound`, `restore`, `isCurator` across `server/.mjs` and `scripts/.mjs`; verified every cited line number myself directly (not from the subagent) — see the commands in §7.
- **Evidence — the autonomy-coupling half is real and live:** `ceilingFor()` (`edge-trust.mjs:150-162`) maps evidence state directly to an authority tier (`QUARANTINED` → `simulation_only`, `CONTRADICTED` → `withheld`), and `authorityFor()` (`edge-trust.mjs:388-400`) is genuinely called from `reEvaluateAllPlans()` (`server.mjs:4294`), itself run at boot (`server.mjs:4315`) and every 24h (`server.mjs:4316-4317`, confirmed `24 60 60 1000`), gating a real auto-email side effect. A blocked attempt is appended to the same* persisted record (`edge-trust.mjs:405-411`), not a separately-desyncable audit store. `CURATORS/isCurator()` (`server.mjs:637-638`) is a real, wired human-permission boundary gating six distinct endpoints (`server.mjs:4365` onward), confirmed by direct grep.
- **Evidence — the human-closure half is not reachable in production:** `confirmConfound()` and `restore()` (`edge-trust.mjs:352-368`) are the *only* way to clear a `QUARANTINED` or `CONTRADICTED` edge. `grep -rn "\.restore(\|\.confirmConfound("` across every server/script file returns **zero hits outside `edge-trust.mjs` and its own test file**. There is no route, script, or curator-UI action that calls either function. An edge that locks to `QUARANTINED` today has no operational path back except direct edits to the raw JSON state file on the production host.
- **Falsification condition:** would be refuted by finding a real route/script calling `confirmConfound`/`restore`, or by finding the KB curator boundary unenforced.
- **Verdict: confirmed on both halves — the mechanism works, and its human-closure path is missing.** Scored 3: the rubric's own anchor for 3 (*"an override path exists but wasn't exercised during the audit"*) is generous here — this override path doesn't just go unexercised, it has no operational entry point at all for the mechanism that most needs one. I didn't score lower because a real, wired, exercised human-permission boundary *does* exist elsewhere in the same file's neighborhood (the curator gate), so "no working human-override path found anywhere" (the 1-anchor) would overstate the gap.

5. Stress Test Results (Stage 5)

No live perturbation was injected against the production service — see §2. What follows is: (a) one real production stress event already on record, read directly rather than re-described from memory, and (b) direct code/test evidence for how each axis is designed to behave, with test-coverage gaps called out explicitly rather than assumed away.

5.1 Latency axis

- **Perturbation:** a real 2026-08-10 incident — the Anthropic account hit a zero-credit-balance state, and separately, two real ~20s OpenRouter timeouts occurred with no clear platform-wide cause (`fallback.mjs:6-44`, `268-281`), documented from the actual investigation, including the hypotheses ruled out: prompt size, a platform incident, host contention, a stale keep-alive socket).
- **Before:** the advisor had a hard dependency on Anthropic's billing state — *"every AI feature returned 400... only the model-backed features died"* (`fallback.mjs:9-11`).
- **After:** OpenRouter became the real primary (not a degraded fallback), Anthropic and DGX-Ollama retained as real fallbacks, one retry added before falling through the ladder, per-rung timeouts tuned to stay under Cloudflare's ~100s tunnel default (`fallback.mjs:62-68`).
- **Classification: FRAGILE-LEANING**, on the system's own terms — this was a well-reasoned, evidence-based **human** fix (a real incident, real investigation, real patch), not the software detecting and adapting on its own. And the patched code itself has close to zero regression coverage: `grep` across every `tests/*.test.mjs` file finds real test coverage for exactly one of `fallback.mjs`'s six exported functions (`unwrapEnvelopedReply`, in `tests/unwrap-enveloped-reply.test.mjs`), `isFallbackWorthy()`, `localAnswer()`, `localChatJson()`, and `chatViaOpenRouter()` — the actual primary-serving and fallback-triggering logic — have no dedicated test file. A future edit could silently regress the exact incident this file was built to survive, with nothing to catch it.
- **Evidence:** `fallback.mjs` in full; `grep -rln "fallback.mjs" tests/*.test.mjs` → one file, testing one of six functions.

5.2 Stakes axis

- **Perturbation:** routing a chat reply through the degraded deep-fallback model (DGX-Ollama) instead of the primary.
- **Before/after:** `localChatJson()` (`fallback.mjs:211-217`) always returns `actions: []`, regardless of what the degraded model tries to emit — the schema *lets* a model drive UI mutations (`set_points`, `set_species`), and this path categorically refuses to let the degraded model use that power. Photo/trophy scoring refuses to route through local models at all, backed by a real measured number (44.9% MAPE, a known-181 buck scored 61.8) — *"returning a confidently wrong trophy score is worse than returning none"* (`fallback.mjs:47-52`).
- **Classification: ROBUST**, real evidence of an asymmetric guardrail working as designed. This is the strongest single stress-axis finding in the audit — a genuinely correct, explicit, low-stakes/high-stakes distinction, hardcoded rather than left to a model's judgment.
- **Caveat pulling this short of Antifragile:** `localChatJson()`'s `actions: []` guarantee has no dedicated test asserting it holds even if a future prompt change causes the fallback model to try emitting actions. Correct today by direct code inspection; not regression-proof.
- **Evidence:** `fallback.mjs:198-217` (comment + code), `fallback.mjs:46-55` (vision-scoring refusal rationale).

5.3 Exposure axis

- **Perturbation:** a real, already-occurred confounded correlation — `HARVEST_REGIME->MALE_SURVIVAL` is stated in `causal-graph.mjs:59` to have a *negative* mechanism (more harvest pressure → lower male survival), but the live Pearson check came back **positive** (`r = 0.144`, confirmed in the 2026-08-12 production snapshot: `evidence_state: "QUARANTINED"`, `classification: "confounded"`, `confounder_status: "known"`).
- **Before:** `edge-trust.mjs:93-97` hand-registers exactly this confounder ahead of time — *"quota-setting itself plausibly responds to herd health... a reverse-causality story this check's own correlation can't separate from a true causal signal."*

- **After:** `classifyCheck()` (`edge-trust.mjs:124-130`) correctly routed the contradiction-shaped result to `"confounded"` rather than `"contradicting"`, which set `evidence_state` to `QUARANTINED` (not the harsher `CONTRADICTED`) and capped the edge's authority ceiling at `simulation_only` (`edge-trust.mjs:157`) — automatically suppressing the auto-email action tied to it, per §4.5.
- **Classification: ROBUST, verging on the "Robust-plus" anchor** — the system correctly avoided both failure directions (blind trust in a noisy signal, and blind dismissal of a real one), and the suppression is a genuine, live, automatic consequence of the evidence state, not a manual patch. It falls short of full "Robust-plus" (and well short of Antifragile) because the discriminating judgment (the confounder registry) was hand-registered by a human *in advance* of this event, not learned *from* it — and per §4.5, the locked `QUARANTINED` state currently has no real path back to a resolved state in production.
- **Important caveat on freshness:** this evidence comes from a frozen snapshot captured 2026-08-12 (one day before this report). I did not pull current production state — that would mean reaching into the live `wydraw.service` host, out of scope for a static/local-repo audit pass (§2). Treat §5.3 as a real, verified *historical* stress event, not a live-as-of-today reading.

6. Remediation Roadmap

PRIORITY	FINDING	FIX	OWNER	STATUS
P0	<code>confirmConfound()</code> / <code>restore()</code> have no operational entry point (§4.5)	Add a real curator-gated route or script that calls them, so a locked edge can actually be resolved without hand-editing production JSON	—	Done, deployed 2026-08-13 — <code>GET /api/edge-trust</code> , <code>POST /api/edge-trust/:source/:target/confirm-confound</code> , <code>POST /api/edge-trust/:source/:target/restore</code> (commit <code>9ff7f79</code>), tested against the real <code>EdgeTrust</code> class (<code>tests/edge-trust-curation-endpoint.test.mjs</code>), verified live in production (<code>curl</code> against <code>drawwise.ai/api/edge-trust</code> returns the expected curator-gated response)
P1	<code>fallback.mjs</code> 's primary-serving/fallback-classification logic is untested (§5.1, §5.2)	Add a test file covering <code>isFallbackWorthy()</code> , <code>localAnswer()</code> , <code>localChatJson()</code> (including the <code>actions: []</code> guarantee), and <code>chatViaOpenRouter()</code>	—	Done, deployed 2026-08-13 — <code>tests/fallback.test.mjs</code> (17 tests), network stubbed via <code>globalThis.fetch</code> per <code>tests/mail-resend.test.mjs</code> 's established convention
P2	No calibration-tracking mechanism ties stated confidence to realized outcomes (§4.4)	Log <code>decisionCardFor()</code> 's <code>confidence</code> field alongside a later-observable outcome, even minimally, so calibration becomes checkable over time	—	Done, deployed 2026-08-13 — <code>server/decision-calibration.mjs</code> , wired into the real <code>SavedPlans</code> lifecycle (<code>/api/plans</code> POST, <code>/advance</code> POST), curator-only <code>GET /api/decision-calibration</code> . Scoped honestly: only <code>APPLY</code> recommendations resolve directly from a single <code>drew</code> report; <code>BANK</code> / <code>WATCH</code> / <code>ABANDON</code> are logged but not force-verdicted
P3	<code>edge-trust.mjs</code> 's authority coupling is wired to exactly one action class	Extend <code>authorityFor()</code> gating to at least one more real automated action, to test whether the pattern generalizes	—	Done, deployed 2026-08-13 — extended the same gate to <code>cohort_pipeline</code> auto-emails via <code>HARVEST_REGIME->MALE_SURVIVAL</code> (the other edge with real evidence, currently <code>QUARANTINED</code> in production), per the code's own pre-existing comment flagging this as ungated

No findings were dropped for time — this list reflects everything this pass's scope actually surfaced, not a sampled subset. Status column added and updated 2026-08-13, verified directly against current code and production rather than assumed from this

document's own earlier text.

Re-audit note: all four remediation items are now closed, each firing this report's own §9 expiry trigger. The §1/§3/§4.5 verdicts (3.45 weighted total, ROBUST) were computed *before every one of these fixes* and understate the current state on at least Guardrail & Agency (§4.5) and Reasoning Depth & Calibration (§4.4). See §8 for additional verified evidence found since the original pass. Treat 3.45/ROBUST as a pre-fix baseline, not current state, until a full re-scoring pass runs against today's code.

7. Provenance Appendix

ID	TYPE	SOURCE	VERIFIED	REFERENCED IN	
E1	source read	<code>server/causal-graph.mjs</code> (full file)	read directly by evaluator	§4.1, §4.2	
E2	source read	<code>server/risk-network.mjs</code> (full file)	read directly by evaluator	§4.3	
E3	source read	<code>server/evidence-ladder.mjs</code> (full file)	read directly by evaluator	§4.4	
E4	source read	<code>server/intervention.mjs</code> (full file)	read directly by evaluator	§4.4	
E5	source read	<code>server/edge-trust.mjs</code> (full file)	read directly by evaluator	§4.5, §5.3	
E6	source read	<code>server/decision-reconciliation.mjs</code> , <code>server/structural-consensus.mjs</code> (full files)	read directly by evaluator	§4.1	
E7	source read	<code>server/fallback.mjs</code> (full file)	read directly by evaluator	§5.1, §5.2	
E8	command output	<code>grep -n</code> for <code>CURATORS</code> , <code>isCurator</code> , <code>authorityFor</code> , <code>checkGraphAgainstData</code> , <code>reEvaluateAllPlans</code> in <code>server/server.mjs</code>	run directly by evaluator, this session	§4.1, §4.5	
E9	command output	<code>`grep -rn ".restore(\`</code>	<code>.confirmConfound("` <code>across server/.mjs</code>, <code>scripts/.mjs`</code></code>	run directly by evaluator, this session	§4.5
E10	test run	<code>npm test</code> — <code>node --test</code> <code>tests/*.test.mjs</code>	run directly by evaluator, this session (not taken from the subagent report): 1549 pass / 0 fail / 0 skipped	throughout	
E11	prior document	internal working notes, 2026-08-12 snapshot (not for public distribution; redacted from this public copy)	frozen snapshot dated 2026-08-12, read by evaluator this session; not independently re-verified against current production state	§5.3	

ID	TYPE	SOURCE	VERIFIED	REFERENCED IN
E12	git log	<code>git log --oneline</code> for <code>edge-trust.mjs</code> , <code>causal-graph.mjs</code> , <code>decision-reconciliation.mjs</code> , <code>structural-consensus.mjs</code> , <code>fallback.mjs</code> , <code>scripts/eval-run.mjs</code>	run directly (partly via subagent, spot-checked by evaluator)	§5.1, §7
E13	subagent finding, independently re-verified	line-number citations for <code>CURATORS</code> , <code>authorityFor()</code> call site, <code>checkGraphAgainstData()</code> , <code>reEvaluateAllPlans()</code> , <code>isCurator()</code>	initially reported by an Explore subagent; every citation used in this report was re-run and confirmed directly by the evaluator before inclusion (commands in E8/E9)	§4.5

8. Addendum – additional verified stress events (2026-08-13)

Mike's independent, patent-level investigation cited eight specific DrawWise incidents as evidence for the architecture's value. Per this report's own §7 gates, none of those claims were entered into the record until verified directly against primary source, this session:

CLAIMED INCIDENT	VERIFIED AGAINST	VERDICT
A known 345-inch bull scored 289 and 299, arithmetic internally consistent	<code>server/reconcile.mjs:3-7</code> : "On Mike's known-345 calibration bull our scoring path reported 289 and 299 — confidently, with a component card. Nothing in the pipeline noticed, because the component sum was internally consistent."	CONFIRMED
A single-reference method produced a -0.82 correlation with known scores	<code>server/reconcile.mjs:20-21</code> : "Single-reference scaling measured -0.82 correlation against known scores on 2026-08-09." — same incident as the row above, the fix (<code>AGREE_PCT</code> / <code>CONFLICT_PCT</code> , structural + terminal comparison) is <code>reconcile.mjs</code> in full	CONFIRMED
Structurally different readings could produce similar totals, concealing recognition errors	<code>server/structural-consensus.mjs</code> 's <code>masked_disagreement</code> field and its own header's IR-photo G4 case (already cited in §4.1 of this report)	CONFIRMED (already in evidence, §4.1)
A $100\times$ scale error survived until dimensional checking exposed it	<code>docs/NDVI_CAUSAL_EDGES.md:101-108</code> : <code>tin</code> 's documented <code>scale_factor: 0.01</code> was never applied — "every <code>tin</code> value ever shipped was 100x too large... verified against the live COG's own rasterio metadata." Fixed; values dropped from ~ 38 – 47 to the correct ~ 0.38 – 0.47 .	CONFIRMED
Provider failures demonstrated that fallback capacity had to be evaluated against the real operating window	<code>server/fallback.mjs</code> 's full incident history — see §5.1 of this report	CONFIRMED (already in evidence, §5.1)
A positive correlation between harvest pressure and male ratio exposed confounding and possible reverse causality	<code>HARVEST_REGIME->MALE_SURVIVAL</code> , <code>edge-trust.mjs:93-97</code> — see §5.3 of this report	CONFIRMED (already in evidence, §5.3)
Unreliable external evidence was withheld from automatic use rather than forced into the reasoning	<code>server/fallback.mjs:46-52</code> : local VLM photo-scoring measured at 44.9% MAPE on Mike's calibration set, excluded from the automated path entirely — "returning a confidently wrong trophy score is worse than returning none"	CONFIRMED
Changed facts caused stored plans to be reconsidered	<code>reconcileWatchedFacts()</code> / <code>reEvaluateAllPlans()</code> — see §4.5 of this report	CONFIRMED (already in evidence, §4.5)

All eight hold up as real, independently-verified incidents, not restated marketing claims. Three (structural masking, provider failures, confounding) were already load-bearing evidence in this report before the addendum; the other five (the 289/299 bull, the -0.82 correlation, the $100\times$ scale error, the withheld-evidence VLM decision, and the changed-facts reconsideration) are newly confirmed here and strengthen — without requiring a score change on their own — the §4.1 (Brittleness) and §5 (Stress Response) findings: this is a system whose real failure history is documented in its own code comments, not hidden or summarized away.

What this addendum does not do: it does not re-run the full six-dimension scoring pass. Combined with the P0–P3 closures logged in §6, this system has moved in two independent ways since the 3.45/ROBUST verdict in §1 —

closed the one confirmed gap this report scored, and gained five newly-verified real stress events. Both push toward a higher score, but neither substitutes for actually re-running §4–§5 against current code. Treat 3.45/ROBUST as stale in the direction described here, not silently corrected.

9. Addendum – a missing dimension: Deployment & Operational Integrity (2026-08-14)

Added one day after this report's original pass, following a real, dated incident this audit's own six dimensions had no way to catch — not a defect in how any of them were scored, but a genuine gap in what the methodology asks at all.

Claim tested: none of Stages 1–6 verify that code *actually running in production* matches what the repo says should be running. They audit reasoning behavior, causal structure, and guardrail wiring — all correctly — but none ask whether a real, merged fix ever reached the host that's supposed to run it.

Real incident, dated and verified: `scripts/mine-usage.mjs` — DrawWise's own real-turn classification pipeline (systemd timer on Canoe, every 3 hours, local-inference classification of every real advisor turn into REAL_BUG/FEATURE_GAP/PROMPT_TUNING/NOTHING_NOTABLE) had a real fix (`d7fac85`, 2026-08-11T19:43:07-04:00) correctly merged to `main`, adding Mike's real account (email redacted from this public copy) to `TARGET_USERS`. The deployed copy on Canoe was never updated — `install-usage-mining-canoe.sh`'s own header states plainly this is a manual, re-run-after-editing step, deliberately kept outside `deploy-canoe.sh`'s automated ship pipeline (to protect the mining cursor/queue state across app rollbacks) — and nobody re-ran it. Confirmed directly (`diff` between the repo copy and the live copy on Canoe, 2026-08-14): the live copy was still missing the fix three full days later, silently blind to virtually all of Mike's real usage in that window, including two real bugs this same product shipped fixes for on 2026-08-13/14 (the Region E/S/W competitive-draw denial, the Area 61 days-per-animal fabrication) — both of which the classifier correctly flagged as REAL_BUG the moment the real fix was actually redeployed.

This audit itself ran inside that blind spot. §1's cover states the evaluated commit as `c2ca0b1` (2026-08-13T14:55:37-04:00) — 43 hours after the un-deployed fix, two days into the exact incident described above — and correctly did not flag it, because `mine-usage.mjs` was never in this audit's declared scope (§2). That is not a scoring miss; it is the methodology having no dimension that would have asked the question even if the file had been in scope.

Falsification condition: would be refuted by finding an existing stage/axis in §3's Score Card that already checks repo-vs-deployed drift for auxiliary services outside the main deploy pipeline. None does — Stage 1 (Brittleness/Component Inventory) checks whether code exists and is wired *in the repo*; Stage 5's three stress axes (Latency/Stakes/Exposure) test runtime behavior under real-time conditions; none test whether the deployed artifact is current.

Verdict: confirmed real gap. Proposed as a 7th top-level dimension — **Deployment & Operational Integrity** — scored on: (a) does every component with a real fix path (repo commit → live effect) have an automated, verified deploy step, or does it rely on someone remembering to re-run a script; (b) is there any drift-detection between repo HEAD and what's actually running, for every deployed component, not just the primary application; (c) if a component is deliberately excluded from the primary deploy pipeline for a real, stated reason (as `mine-usage.mjs` explicitly is), does that exclusion carry a substitute integrity check, or does the exclusion itself become the blind spot.

Not scored into §3's 3.45 total — same convention §6/§8 already established for a dated finding that pushes the methodology forward without retroactively re-scoring a subsystem outside its own reach: this dimension is a whole-system property, evaluated here only because DrawWise happened to be the system where the gap was caught, not because it belongs to the causal-reasoning subsystem's own §2 scope.

Where this should actually live: this report is one filled instance of a generic, reusable audit methodology used across other engagements (per this document's own opening line). The rubric-level template with the granular 6-dimension scoring isn't in this repo; the public description of the methodology ([theonedegreedispatch.com, /antifragility-audit.html](https://theonedegreedispatch.com/antifragility-audit.html)) is process-level, not rubric-level. This dimension belongs in that master template, not just this one filled report — otherwise every other system audited under this methodology carries the identical blind spot.

10. Certification

This report reflects direct testing against DrawWise's causal-reasoning subsystem at commit `c2ca0b1` (HEAD, 2026-08-13), conducted 2026-08-13 by Claude (Sonnet 5), with **no independent adversarial second review**. It covers the scope defined in §2 and no more — in particular, it does not reflect current (as of report date) production state of `data/edge-trust.json`, and it does not reflect any live model behavior (no costed eval was run).

This report expires — and the audit must be re-run, not just re-issued — upon:

- Any change to `edge-trust.mjs`'s evidence-state/authority-ceiling logic, or to which action classes call `authorityFor()`
- `server/application-cycle.mjs` (uncommitted at audit time) landing and touching any file in scope
- A confirmed production divergence event on any causal-graph edge (a new `QUESTIONED`/`QUARANTINED`/`CONTRADICTED` state)
- Six months out (2027-02-13), sooner if this subsystem's action-class coverage expands per the P3 remediation item
- Any wiring of `confirmConfound()`/`restore()`/`resolvePendingReview()` into a real route or script (would directly change the §4.5 and §6 verdicts, likely upward)

Signed: Claude (Sonnet 5) • **Adversarial review:** none conducted • **Date:** 2026-08-13

Methodology Note

Produced using the One Degree Antifragility Audit methodology (<https://theonedegreedispatch.com/antifragility-audit>). Methodology and tooling: PATENT PENDING | U.S. Provisional Patent Application No. 64/132,274. Tool licensed under Apache License 2.0 — see LICENSE and NOTICE in the accompanying skill package.

Generated by the One Degree Antifragility Audit methodology — theonedegreedispatch.com/antifragility-audit. Every finding traces to a real file, line, commit, or test — re-check any of them directly. Methodology and tooling: **PATENT PENDING | U.S. Provisional Patent Application No. 64/132,274**. Tool licensed under Apache License 2.0 — see LICENSE and NOTICE.