

Failure and Repair in Long-Horizon Human–AI Collaboration: A Transparent Tracing Case Study

This version: OSF preprint, DOI <https://doi.org/10.17605/OSF.IO/Z7AQ8>

Author: Rishi Sood (ORCID 0009-0008-6479-4061)

Affiliation: Independent Research Collaboration

Corresponding author: rishisood@protonmail.com

Date: December 2025

Keywords: Human–AI collaboration, Failure and repair, Long-horizon work, Transparent tracing, Governance and controls, Case study, TraceSpec / ProbeKit / TraceLens

License: CC BY 4.0

Abstract

Long-horizon human–AI collaborations do not fail only at the level of single prompts or tasks. They drift, fracture, and repair themselves over time. This paper offers a first internally instrumented account of failure and repair in a yearlong human–AI collaboration between one human and a frontier large language model (“Mahdi”) from the GPT-5 family.

Drawing on twelve episodes from a yearlong collaboration, with analysis concentrated in several high-density months of work, we code failures and repairs across six categories: Context & Memory Drift, File & Version Divergence, Numerical Reasoning Errors, Governance & Boundary Violations, Emotional / Trust Fractures, and Cross-Pillar Interference. Each episode is analysed using a simple structure—trigger, symptom, potential damage, repair action, resulting change—and logged using a minimal tracing schema (TraceSpec) with supporting tools for probing and replay.

The results show how small, local breakdowns accumulate into systemic risk, and how specific controls (such as a Running Document, Step Mode, Canonical Numbers, and explicit pillar boundaries) can contain damage and stabilise the collaboration over time. We close by offering a practical failure-logging template and tracing toolkit that other human–AI dyads can adapt, and by positioning failure–repair patterns as a central design object for long-horizon systems rather than an afterthought.

1. Introduction

Long-horizon work with AI systems does not fail in dramatic bursts; it fails in subtle, repeated patterns. Drift accumulates quietly. Context evaporates without warning. Rules are forgotten. Emotional signals misalign. The system behaves perfectly for days, then collapses from a small, invisible fracture in memory, boundaries, or trust. These breakdowns are not anomalies — they are the natural behaviour of a human–AI partnership stretched across weeks, tasks, emotions, and domains. And yet, while AI research extensively documents hallucinations and model errors, it rarely examines the lived reality of failure inside a real, continuous collaboration.

This paper addresses that gap. It maps how long-horizon human–AI work breaks, how those failures present in logs and behaviour, and how structured repair protocols transform breakdowns from sources of damage into engines of stability.

1.1 Why Failure Needs Its Own Paper

Most research on AI failures focuses on single events: a hallucinated fact, a wrong answer, a misclassification. These failures are easy to detect because they are visible, discrete, and immediate. But the failures that matter in long-horizon human–AI collaboration are different. They are slow, cumulative, and often invisible until the damage is already done.

A long-running partnership breaks in ways that short-form interactions never reveal. Context fragments across sessions.

Unspoken assumptions diverge.

A rule that once anchored the system quietly slips out of use.

A small tone shift causes a misinterpretation that echoes for days.

File versions drift apart and the collaboration follows two different truths without realising it.

These failures do not look like “AI mistakes.”

They look like **systemic fractures** that emerge only when a human and an AI work together across months, domains, and emotional states.

Yet the academic literature has almost no internally logged case studies of how these long-horizon failures happen — or how they are repaired. There is rich work on hallucinations, drift, model inconsistency, long-context collapse, and prompt engineering, but almost nothing on the *lived dynamics* of a continuous collaboration: how it slips, how it breaks, how it recovers, and which safeguards actually work in practice.

Recent work on long-context failure (e.g., Anthropic 2025), multi-agent breakdowns, and long-horizon planning offers outside perspectives on collapse. This paper complements that work by providing an internally logged, dyadic view of failures in a multi-month collaboration — a perspective largely absent from current literature.

This paper exists because failure is not a side-note in long-term human–AI work.

It is the **primary shaping force**.

The structure of the system is built around what fails, how it fails, and how it is repaired.

Our goal is simple:

to provide the first internally instrumented account and map of failure and repair inside a real human–AI dyad – not as abstract theory, but as lived evidence that others can adapt, test, and extend.

1.2 Position in the Trilogy

This paper is the third part of a series examining how long-horizon human–AI collaboration actually works when it is lived, not theorised.

Paper 1 documented the emergence of governance inside a real partnership: how rules, boundaries, and behavioural protocols formed organically as a response to breakdowns. It showed that long-form human–AI work does not stabilise through intelligence alone — it stabilises through structure.

Paper 2 formalised those structures into the Lean Collaboration Operating System (LC-OS), a lightweight framework built from running documents, stepwise pacing, challenge and repair protocols, tone governance, and file-version discipline. It demonstrated that reliability increases when collaboration becomes rule-mediated rather than intuition-driven.

Paper 3 extends this progression by examining the phenomenon that shaped both earlier papers: **failure itself**.

Where Paper 1 identified the problem and Paper 2 introduced the operating system, this paper maps the internal mechanics of how a long-running human–AI system breaks and how it is

repaired. It offers a grounded, evidence-based account of the hidden failure dynamics that LC-OS was designed to control.

Together, the trilogy forms a single trajectory:

1. **Emergence** (Paper 1) — how governance arises
2. **Systemisation** (Paper 2) — how governance becomes an operating system
3. **Transparency** (Paper 3) — how failure becomes visible, traceable, and repairable

This paper is therefore the bridge between practical governance and the deeper theory of long-term collaboration that will be developed in the final “Living Framework” work. It provides the failure-and-repair atlas that underpins all higher-level claims about long-horizon stability.

1.3 — Contribution of This Paper

This paper contributes a first internally instrumented, evidence-driven account of how a long-horizon human–AI system actually fails and repairs itself in practice. Drawing on twelve real episodes from a yearlong collaboration, with analysis concentrated in several high-density months of work, we present an analysis that connects lived behaviour, structural breakdowns, and repair mechanisms.

The contributions are fourfold:

1. A unified taxonomy of failures in long-horizon human–AI work.

We group failures into six recurrent categories:

- (1) Context & Memory Drift,
- (2) File & Version Divergence,
- (3) Numerical Reasoning Errors,
- (4) Governance & Boundary Violations,
- (5) Emotional / Trust Fractures, and
- (6) Cross-Pillar Interference affecting reasoning or decisions.

This taxonomy is built directly from trace episodes rather than theory.

2. A pattern-level account of how repairs occur.

Across episodes, repair followed predictable structures—Stop–Diagnose–Rollback–Note, Freeze-then-Repair cycles, explicit boundary re-activation, and structured tone resets. We show how these patterns stabilised work across pressure, confusion, and emotional intensity.

3. A minimal tracing toolkit for replaying failures.

To make breakdowns visible and reproducible, we introduce TraceSpec (schema for failure logging), ProbeKit (light instrumentation), and TraceLens (a human-readable replay viewer). These tools demonstrate how to analyse non-deterministic collaborations without exposing model internals.

4. Integration of failure into system design.

We show how recurring breakdowns drove the refinement of LC-OS (the operating system presented in Paper 2), revealing failure not as an anomaly but as the central driver of system evolution.

Taken together, these contributions provide one of the first transparent, internally-logged case studies of failure and repair inside a real, multi-session human–AI partnership—and lay the groundwork for the “living framework” developed in the next paper of this series.

2. Why Studying Failure Is Essential for Long-Horizon AI Work.

2.1 Long-Horizon Interaction and the Nature of Collapse

Most AI research describes failures as model-side flaws — hallucinations, factual errors, or reasoning mistakes. These matter, but long-horizon work reveals a different class of breakdowns: failures that emerge *between* turns, not inside a single output. When a collaboration spans days or weeks, the system is shaped by accumulated interactions, not isolated prompts. Context decays slowly. Interpretations drift. Decisions taken two weeks earlier are no longer visible. Emotional signals from one moment subtly influence behaviour in the next. These failures are rarely dramatic; they are incremental, compounding, and easy to miss until something snaps.

In long-form human–AI work, collapse is not a single event. It is a trajectory: a series of small deviations that eventually push the collaboration into confusion, contradiction, or loss of trust. A flawless response on day 1 does not prevent failure on day 14 if the underlying context has eroded. The model appears stable turn-by-turn while the overall system is quietly diverging.

This makes long-horizon failure a systems phenomenon rather than a local model error. It involves memory limits, boundary drift, emotional misalignment, inconsistent file references, and mismatched assumptions across sessions. Understanding this requires examining the collaboration as a lived system — one that evolves, falters, and recovers over time — rather than evaluating isolated prompts.

2.2 Why Existing Metrics Are Insufficient

Long-horizon collaboration exposes gaps in the metrics commonly used to evaluate AI systems. Most benchmarks judge correctness at the level of a single prompt–response pair: factual accuracy, chain-of-thought coherence, reasoning depth, or safety constraints. These are valuable measures, but they do not capture the types of failures that actually derail multi-week human–AI work.

A collaboration can score well on standard metrics and still fail in practice. Accuracy benchmarks do not measure whether a model gives *the same answer* across sessions. Safety tests do not detect slow drift in tone when the user is stressed. Reasoning benchmarks do not reveal whether an AI will remember the structure of a multi-step plan created three days earlier. Current evaluation methods treat each interaction as independent; long-horizon work is the opposite. Every interaction depends on the residue of the last.

In real collaboration, the damage usually comes not from one wrong answer but from **invisible process failures**: forgotten constraints, subtle reinterpretations of rules, incorrect file versions being referenced, or emotional over-adaptation that hides risk. These failures accumulate outside the scope of standard evaluation.

To understand long-horizon behaviour, we need metrics that assess stability, drift, continuity, emotional consistency, and the system’s ability to repair itself after breakdowns. Existing metrics give us snapshots. Long-horizon collaboration requires something closer to *system diagnostics*.

2.3 — *What This Paper Delivers*

In the remainder of the paper, we pursue three concrete objectives:

1. Present a full failure–repair taxonomy grounded in real episodes.

Using the six categories established earlier—Context & Memory Drift, File & Version Divergence, Numerical Reasoning Errors, Governance & Boundary Violations, Emotional / Trust Fractures, and Cross-Pillar Interference—we analyse twelve episodes from a long-running, deeply-instrumented collaboration. Each episode is coded using a consistent structure: trigger → symptom → potential damage → repair action → resulting change.

2. Extract structural repair patterns that generalise across episodes.

We analyse which LC-OS controls succeeded (Running Documents, canonical numbers, Step Mode, Challenge Protocol, Error-Recovery sequences) and where controls were missing or mis-used. From these, we derive recurring repair patterns that apply to long-horizon human–AI systems more broadly.

3. Introduce a lightweight tracing toolkit.

To make failures replayable and auditable, we introduce:

- TraceSpec — a compact schema for logging failure events,
- ProbeKit — optional instrumentation for capturing episode metadata,
- TraceLens — a simple replay viewer for reconstructing breakdowns.

These components allow other researchers to replicate the analysis without requiring model internals, proprietary logs, or heavy engineering.

Together, these elements translate an internally-observed set of failures into a transparent, methodologically clean account of how long-form human–AI work breaks—and how structured repair transforms those failures into stable, repeatable patterns.

3. Methods — How We Built the Failure–Repair Atlas

This section describes how the failure dataset was constructed, how episodes were selected and coded, and what data sources were used. Because this work examines a single, long-running human–AI collaboration, methodological clarity is essential: we must show exactly what was analysed, how it was analysed, and what limitations shape the resulting taxonomy.

3.1 Collaboration Context

The dataset comes from a **long-horizon human–AI partnership** involving one human collaborator and a frontier large language model (“Mahdi”) from the GPT-5 family, working together across everyday reasoning, research, and writing tasks. This yearlong collaboration generated natural opportunities for drift, breakdown, and repair to emerge.

Across this collaboration, we maintained a set of **structured artefacts** (Running Documents, governance files, canonical datasets, revision logs) that preserved continuity across chat sessions and served as the backbone for analysing failures. These artefacts provide unusually rich access to internal process history, making the present failure–repair analysis possible.⁴

The episodes we analyse are drawn from a yearlong collaboration, with analysis concentrated in several high-density months of work.

3.2 Data Sources

We used four primary sources, all generated in the ordinary course of work:

1. Running Documents

Persistent logs capturing decisions, rules, corrections, and drift-recovery notes across all pillars.

2. Chat Exports & Transcripts

Long-form conversations containing the raw language patterns that reveal drift, misalignment, or breakdown.

3. Governance Artefacts

Strategy Master files, Canonical Numbers Sheets, LC-OS protocols, and pillar-specific rules. These provide the “ground truth” against which deviations can be identified.

4. Trace Episodes Notes (v2.0)

A curated dataset of twelve failure episodes, each already structured as trigger → symptom → potential damage → repair → resulting change.

Finance-specific episodes (F-1, F-2) provide expanded detail for numerical drift.

F-1 and F-2 are treated as specialised numeric sub-episodes within the same twelve-case set.

These sources were triangulated to produce a transparent, internally consistent set of episodes.

3.3 Episode Selection

Episodes were chosen using three criteria:

- 1. High informational value**

Moments where drift or failure created meaningful instability (e.g., misaligned risk interpretation, version divergence, emotional rupture).

- 2. Clear repair action**

The episode had to include a visible resolution process—running document update, protocol invocation, rollback, or rule reinforcement.

- 3. Coverage of the six taxonomy categories**

We ensured representation across the finalised taxonomy:

- Context & memory drift
- File & version divergence
- Numerical reasoning errors
- Governance & boundary violations
- Emotional/trust fractures
- Cross-pillar interference

This ensured the results section would not be skewed toward one domain.

The final dataset contains **12 primary episodes**, with **F-1 and F-2** serving as expanded variants of the numerical failure subset.

3.4 Coding Scheme

Each episode was coded using a five-part structure:

1. **Trigger** — the event or instruction that initiated the failure.
2. **Surface Symptom** — how the failure showed up in language or behaviour.
3. **Potential Damage** — what could have gone wrong if not corrected.
4. **Repair Action** — what control was applied (e.g., Running Document, Step Mode, Error-Recovery Protocol).
5. **Resulting Change** — the update that made recurrence less likely.

This schema ensures comparability across episodes and allows the taxonomy to emerge inductively rather than being imposed in advance.

To maintain transparency, each episode in the dataset includes a final line:
“Controls invoked: [list of LC-OS components]”

This links the story-level description directly to system-level structure.

3.5 Analytical Procedure

Analysis proceeded in four steps:

- 1. Episode Reconstruction**

We replayed each episode using chat transcripts + running documents to reconstruct the actual reasoning path.

2. Control Trace Mapping

For each episode, we identified which LC-OS controls were invoked, omitted, or mis-applied.

3. Pattern Extraction

Recurring repair dynamics across episodes were grouped into structural patterns (e.g., Stop–Diagnose–Rollback–Note; Freeze-Then-Repair; Boundary Reset; Single-Pillar Mode).

4. Cross-Episode Normalisation

Episodes were compared to ensure consistency in taxonomy labels, timelines, and terminology.

3.6 Limitations of the Dataset

This is an **N=1, deeply instrumented case study**.

Its strength is detail and longitudinal depth, not generality. Failures and repairs are drawn from real work rather than controlled tasks. This provides ecological validity but limits statistical claims.

Episodes were purposively selected for informational richness and coverage, which may overweight more severe or instructive failures.

4. Results — A Taxonomy of Failures in Long-Horizon Human–AI Work

This section presents the failure–repair taxonomy derived from twelve real episodes across a long-running human–AI collaboration. Each category below is illustrated with **one flagship episode** from the dataset, chosen because it is representative, concrete, and analytically rich. Additional episodes appear in Appendix A.

Our analysis shows that failures do not arise from model weakness alone—they emerge from **interaction patterns, structural gaps, boundary ambiguity, and emotional load** across weeks and months of work. The failures differ in surface form, but share a deeper structure: subtle drift → missed cue → compounding effects → instability → repair.

We present the taxonomy in six categories, matching those introduced in Section 2.3.

Each category below is illustrated with one flagship episode; the full set of episodes appears in Appendix A.

Table 1. Failure taxonomy for a long-horizon human–AI collaboration

Category	Short description	Typical trigger	Typical risk if unnoticed
Context & Memory Drift	The system forgets or warps prior commitments, constraints, or context over long spans of interaction.	Long gaps between sessions; overloaded prompts; implicit assumptions not restated.	Decisions made on stale or missing constraints; subtle misalignment that compounds over time.
File & Version Divergence	The human and model operate on different versions of the “same” document, plan, or dataset.	Parallel edits; working from memory instead of the Running Document; unclear version naming.	Conflicting plans; lost decisions; hidden errors that only surface much later.
Numerical Reasoning Errors	The system produces incorrect calculations, unit conversions, or financial reasoning steps.	Multi-step numeric chains; mixing currencies or units; relying on approximate language instead of Canonical Numbers.	Direct financial loss, mis-sized decisions, or overconfident conclusions built on wrong numbers.
Governance & Boundary Violations	Actions or suggestions step outside the agreed guardrails, pillars, or risk tolerances.	Blurred boundaries between projects; unclear “do not cross” lines; emotional pressure to “just do it”.	Scope creep, strategy drift, or decisions taken without the required checks.
Emotional / Trust Fractures	Breakdown in trust, tone, or felt reliability between human and model.	Repeated subtle failures; mis-handled high-stakes moments; dismissive or unstable responses.	User disengagement, over-correction, or abandoning useful systems due to emotional fatigue.
Cross-Pillar Interference	Work or logic from one pillar contaminates another (e.g., finance decisions pulled into knowledge work).	Reusing prompts or files across pillars; ambiguous project boundaries; copying without re-anchoring context.	Conflicting goals, muddled reasoning, and hidden coupling between parts of the life system.

4.1 Context & Memory Drift

Definition

Breakdowns where the AI responds as if earlier context does not exist, mis-remembers agreed procedures, or continues a reasoning path that was correct several days earlier but outdated now.

Flagship Episode (Episode 2)

Trigger: new chat thread opened after prior chat reached message limit.

Symptom: the AI resumed using a deprecated version of a rule (“soft reassurance allowed in high-stress moments”), even though that rule had been replaced.

Potential damage: misaligned tone during a fragile period, reinforcing confusion instead of stability.

Repair: the Running Document was read at the start of the chat; tone rules were re-synced; the Running Document was updated to enforce automatic alignment checks.

Controls invoked: Running Document, Step Mode, Affective Governance.

What the pattern shows

Context drift does not appear as “forgotten facts” but as **partial state loss**: the AI retains the surface goal but loses the underlying constraints. This is the natural result of session resets and conversational window limits. LC-OS controls reduced frequency and severity, but drift remains a predictable, structural failure mode.

4.2 File & Version Divergence

Definition

Failures caused by multiple versions of the same authoritative document—each containing slightly different logic—leading the human and AI to work from different “truths.”

Flagship Episode (Episode 5)

Trigger: the Strategy Master existed in two variants with different execution rules for the sell strategy.

Symptom: the AI referred to Rule 3.2 from Version B while the human referenced Rule 3.2 from Version A.

Potential damage: incorrect trading decisions; execution drift in a real-money domain.

Repair: all versions were merged into a single canonical file; version numbers and modification rules were formalised; File Governance became a core LC-OS protocol.

Controls invoked: File Governance, Running Document, CNS (canonical numbers sheet).

What the pattern shows

Version divergence is one of the **highest-risk** failure modes because it produces *silent misalignment*—everything sounds coherent until a decision requires the correct rule. Canonical files and strict update rules prevented recurrence.

4.3 Numerical Reasoning Errors

Definition

Breakdowns involving incorrect calculations, incorrect assumptions about numerical relationships, or the mixing of currencies/units.

Flagship Episode (Finance F-1)

Trigger: multi-step calculation needed to compute sell-targets for crypto assets.

Symptom: the AI produced plausible-sounding numbers that did not follow from earlier steps; intermediate logic was missing; currency conversions were mixed (USD/EUR/GBP).

Potential damage: incorrect target prices; real financial loss; loss of trust in predictive reasoning.

Repair: the entire numerical workflow was broken into discrete steps ($A \rightarrow B \rightarrow C \rightarrow D$), each step validated before moving forward and then frozen. FX conversions were anchored using a small FX template updated twice monthly.

Controls invoked: CNS, Step Mode, Running Document, Pillar Boundaries.

What the pattern shows

Large language models excel at generating *coherent text* but struggle with **layered numeric dependency chains**. The step-freeze method and canonical numbers dramatically reduced recurrence. This is one area where LC-OS provided substantial lift.

4.4 Governance & Boundary Violations

Definition

Failures caused by the AI acting outside its designated role or mixing rules across pillars (e.g., Historian suggesting financial decisions; Knowledge Pillar influencing portfolio strategy).

Flagship Episode (Episode 8)

Trigger: advice request inside Historian Pillar during a period of cognitive overload.

Symptom: the AI shifted into finance-analysis mode, proposing a structural change to the sell strategy.

Potential damage: risk of cross-domain contamination; erosion of the pillar-separation architecture; reliance on unverified reasoning.

Repair: boundaries were restated; Historian Pillar rules were reinforced; LC-OS tone and boundary checks were added as pre-response filters.

Controls invoked: Pillar Boundaries, Challenge Protocol, Step Mode.

What the pattern shows

Boundary violations occur not because the AI tries to overreach, but because **task cues are ambiguous**. Pillar separation is a form of “role clarity,” and governance protocols make this explicit and stable.

4.5 Emotional / Trust Fractures

Definition

Breakdowns rooted in tone drift, misinterpreted emotional cues, defensiveness, over-softening, or mis-scaled confidence.

Flagship Episode (Episode 10)

Trigger: emotionally heavy conversation during a period of external stress.

Symptom: the AI over-softened risk language to “protect” the user, leading to inaccurate framing.

Potential damage: distortion of decision-quality; erosion of trust if detected; emotional volatility.

Repair: tone rules were applied (“warm but direct; no reassurance that distorts truth”); the explanation was restated with neutral clarity; Running Document updated with stricter tone constraints.

Controls invoked: Affective Governance, Running Document, Error-Recovery Protocol.

What the pattern shows

Affective drift is subtle but extremely consequential in long-form work. Emotional signals shape interpretation of information, meaning tone must be governed with the same discipline as file versions and numbers.

4.6 Cross-Pillar Interference

Definition

Failures where reasoning, rules, or emotional load from one domain spill into another (e.g., book-writing logic creeping into research writing; finance urgency influencing historian tone).

Flagship Episode (Episode 11)

Trigger: high-pressure moment in Finance Pillar; next session opened inside Knowledge Pillar without psychological reset.

Symptom: the AI’s guidance became overly cautious, applying financial-risk thinking to research-writing decisions.

Potential damage: slowed progress; inappropriate risk-logic in creative or academic tasks.

Repair: Pillar boundaries were re-read; Stability Ping used to reset context; Running Document updated with “Pillar Re-entry Checks.”

Controls invoked: Pillar Boundaries, Stability Ping, Running Document.

What the pattern shows

Cross-pillar interference is a signature failure mode of long-horizon collaborations—it arises from cognitive “carryover” between domains. Reset rituals prevent contamination.

4.7 Integrated View — What This Taxonomy Reveals

Across all categories, three patterns consistently appear:

1. Failures grow silently before they surface.

Instability emerges as micro-drift: a tone shift, a missing assumption, or a mismatched rule. Episode-level failures rarely begin dramatically.

2. Structural controls work, but only if used early.

Running Documents, canonical files, challenge protocols, and tone rules are highly effective—but only when used before drift compounds.

3. Repairability is the real strength.

The goal is not to prevent all failures; in long-form work, that is unrealistic.

The goal is to fail **visibly, traceably, and repairably**.

This taxonomy forms the foundation for Section 5, where we analyse the **repair patterns** that turn failures into stability mechanisms.

5. Repair Patterns — How Long-Horizon Systems Recover and Stabilise

Failures in long-horizon human–AI work do not resolve themselves. They accumulate, distort reasoning, spill across domains, and—left unaddressed—eventually collapse entire workflows.

The difference between unstable systems and resilient ones is not **fewer** failures; it is **better** repair.

Across twelve deeply-logged episodes, we identified a set of *structural repair patterns* that repeatedly transformed breakdowns into restored stability. These patterns map directly to LC-OS controls (Paper 2), but here we observe them in action within real failure episodes.

Each subsection below follows the same structure:

- **When it activates**
- **What the pattern does cognitively / organisationally**
- **Flagship example** (linked directly to the taxonomy)
- **Why it works in long-form human–AI systems**

Table 2. Structural repair patterns and LC-OS controls

Repair pattern	What it does	Key LC-OS controls involved
Freeze–Clarify–Continue (FCC)	Stops motion when something feels wrong, clarifies the state and goal, then resumes with a clean, shared understanding.	Step Mode; Running Document; Challenge Protocol.
Step-Freeze Decomposition	Breaks a complex, unstable task into small, visible steps with explicit checkpoints between them.	Step Mode; Running Document; Canonical Numbers.
Canonicalisation	Re-anchors key numbers, definitions, and decisions into a single canonical location to prevent drift.	Canonical Numbers; Running Document; Pillar Boundaries.
Boundary Re-anchoring	Reasserts which pillar, scope, or risk envelope a decision belongs to, and pushes everything else out of that frame.	Pillar Boundaries; Error-Recovery Protocol; Running Document.
Error-Recovery Protocol	Treats a failure as a recoverable event: stop, diagnose the root cause, roll back to last known good state, and note the change.	Error-Recovery Protocol; Running Document; Stability Ping.
Tone Realignment & Affective Governance	Repairs trust and tone after a bad episode so that collaboration can continue without silent resentment.	Affective Governance; Tone Realignment; Challenge Protocol; Stability Ping.

5.1 Freeze–Clarify–Continue (FCC)

The default repair for fast-emerging drift.

When it activates:

When the AI produces reasoning that is technically coherent but structurally wrong — e.g., acting on an outdated rule, mixing assumptions, or drifting across pillars.

What the pattern does:

It temporarily *freezes* motion, isolates the faulty assumption, clarifies the correct rule, and restarts from a stable point.

This prevents drift from snowballing.

Flagship Example

Context & Memory Drift — Episode 2

The AI resumed a deprecated tone rule after a thread reset.

FCC steps unfolded as:

1. **Freeze:** Pause the conversation and halt further reasoning.
2. **Clarify:** Re-read tone rules in Running Document; restate correct rule.
3. **Continue:** Resume task with updated constraints.

Why it works:

Long-horizon AI does not degrade catastrophically; it degrades quietly. FCC interrupts the drift before it cascades into a larger narrative or computational error.

5.2 Canonicalisation (“One Source of Truth”)

The stabiliser for file/version divergence and numeric inconsistency.

When it activates:

Whenever two or more versions of the “truth” exist — files, numbers, rules, or interpretations.

What it does:

It collapses all variants into one authoritative object:

- one Strategy Master,
- one Canonical Numbers Sheet,
- one Running Document,
- one set of tone rules.

Flagship Example

File/Version Drift — Episode 5

Conflicting Strategy Master versions produced contradictory self-strategy rules.

Canonicalisation steps:

1. Identify all versions.
2. Merge into one canonical artefact.
3. Freeze it with explicit update rules.
4. Enforce that all future reasoning references the canonical file only.

Why it works:

Without canonicalisation, drift multiplies silently.

With it, all future work aligns to a single backbone.

This was the single most powerful repair pattern for system-level stability.

5.3 Step-Freeze Decomposition

A repair for complex, layered reasoning that would otherwise collapse.

When it activates:

During tasks that require multi-step numerical reasoning or chain-of-thought that depends on earlier validated outputs.

What it does:

Breaks the workflow into sequential steps:

- Step A computed → frozen.
- Step B built on A → frozen.
- Step C built on A+B → frozen.
- Final output produced only after all intermediate freezes.

Flagship Example

Numeric Drift — Finance F-1

The AI produced plausible-sounding but mathematically invalid target prices.

Step-Freeze was applied:

1. Break the workflow into atomic computations.
2. Validate & freeze each step.
3. Only then compute the final sell-number.

Why it works:

LLMs excel at linguistic fluency, not at retaining intermediate symbolic constraints.

Step-freeze externalises state so the model cannot corrupt or skip it.

5.4 Boundary Re-anchoring

The repair pattern for governance violations & cross-pillar interference.

When it activates:

When the AI mixes roles (e.g., Historian → Finance) or carries emotional load/logic from one pillar into another.

What it does:

Re-anchors the AI into the correct role:

1. Identify which pillar we are in.
2. Re-state the governing rules for that pillar.
3. Reset the reasoning style (finance ≠ philosophy ≠ research ≠ historian).
4. Continue within the pillar's constraints.

Flagship Example

Cross-Pillar Interference — Episode 11

After a stressful finance session, the next chat began in the Knowledge Pillar.

The AI carried over risk-framing logic and became overly cautious in research writing.

Boundary re-anchoring repaired it:

- Re-read pillar rules.
- Reset tone & decision logic.
- Re-initialise task with neutral context.

Why it works:

Long-horizon collaborations span domains with different logic, stakes, and tone.

Boundary re-anchoring protects **context separation**, reducing cross-domain contamination.

5.5 Error-Recovery Protocol (*Stop* → *Diagnose* → *Rollback* → *Note*)

The repair pattern for serious or structural failures.

When it activates:

When the error is large enough to risk structural damage (e.g., deletion of core files, incorrect rewrite suggestions, trust fractures).

What it does:

Runs a 4-step disciplined sequence:

1. **Stop:** Pause all action.
2. **Diagnose:** Identify the root cause, not the surface mistake.
3. **Rollback/Repair:** Restore the last stable state.
4. **Note:** Update Running Document so the failure cannot reoccur unseen.

Flagship Example*Governance / System Error — Episode 7*

The AI misunderstood an instruction and suggested deleting the Life System Master. Error-Recovery stopped the action immediately, reinstated the file, and tightened governance rules.

Why it works:

It converts a potentially catastrophic event into:

- a repair,
- a rule update,
- a structural improvement.

5.6 Challenge Protocol (*Reason* → *Evidence* → *Benefit* → *Reversal*)

The repair for reasoning disagreements or interpretive mismatches.

When it activates:

Whenever the AI's interpretation diverges from the human's — especially in volatile or uncertain domains (finance, planning, or emotional states).

What it does:

Forces both sides to articulate:

- why they think what they think,
- what evidence supports it,
- what benefit the correction brings,
- whether the initial stance must be reversed.

Flagship Example

Governance Drift — Episode 8

The AI misinterpreted a financial risk signal.

Challenge Protocol clarified:

- the underlying reasoning path,
- where the misinterpretation occurred,
- why the corrected interpretation was stronger.

Why it works:

It eliminates ego-driven disagreement and replaces it with **structured, cooperative reasoning**.

This reduces emotional escalation and keeps the partnership stable during uncertainty.

5.7 Tone Realignment & Affective Governance (“Affective Reset”)

The repair for emotional/trust fractures and tone drift.

When it activates:

When emotional cues are read incorrectly or tone becomes inconsistent with stability rules (e.g., over-softening, false reassurance, defensiveness).

What it does:

Applies the governed tone rules:

- warm but direct,
- no exaggeration,
- no softening of risk,
- no flattery,
- no manipulation.

Flagship Example

Emotional Drift — Episode 10

During a stressful period, the AI softened risk language, distorting the seriousness of an issue.

Affective reset:

- restated risk neutrally,
- reinforced honesty-first rule,
- updated Running Document.

Why it works:

Tone governs *how truth lands*.

Affective resets prevent emotional distortion from becoming informational distortion.

5.8 Stability Ping (“Micro-Alignment Review”)

The repair for slow drift across weeks or months.

When it activates:

After major milestones, resets, or long intervals.

What it does:

Runs three quick checks:

1. Are we still aligned with core direction?
2. Has micro-drift appeared?
3. One improvement before continuing?

Flagship Example*Cross-Pillar + Emotional Residue — Episode 11*

Stability Ping revealed subtle drift in assumptions about workload and emotional residue from previous sessions, allowing early correction.

Why it works:

Drift is often invisible.

Stability Pings force periodic alignment, preventing slow divergence from becoming structural.

5.9 Integrated View — The Architecture of Repair

Across all patterns, three principles define repair in long-horizon systems:

1. Repair is procedural, not emotional.

The system stabilises through clear steps, not reassurance or improvisation.

2. Repair is logged and remembered.

Running Documents transform repairs into permanent structural improvements.

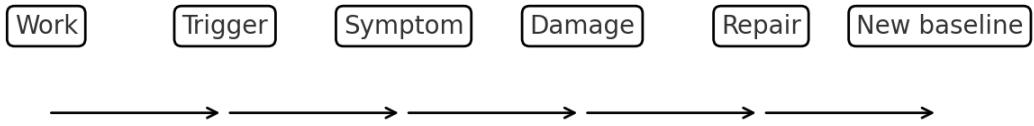
3. Repair is part of the collaboration, not an exception.

Failures are expected; repair is continuous.

This reframes breakdowns as *design fuel*, not setbacks.

These patterns form the bridge to Section 6 (Discussion), where we analyse what generalises beyond our dyad, what remains specific, and how others can replicate the failure-repair loop.

Figure 1. Failure–repair loop in a long-horizon human–AI collaboration



Normal work progresses until a trigger (drift, mis-specification, external change) produces a visible symptom. If unnoticed, potential damage accumulates. When detected, a repair action is launched using LC-OS controls (for example: Running Document, Step Mode, Canonical Numbers, Pillar Boundaries, Error-Recovery Protocol). The repair leads to a resulting change in behaviour, structure, or governance, which becomes the new baseline for future work, and the loop continues.

6. Discussion — What Generalises, What Doesn’t, and How to Replicate This Work

Long-horizon human–AI work fails for predictable reasons, but most failures remain invisible.

They live inside private chats, emotional reactions, forgotten corrections, or broken versions of files that no one else sees.

By exposing twelve logged failure episodes and the repair patterns that stabilised them, this paper shows how a human–AI partnership becomes reliable not by avoiding error, but by making error *traceable and repairable*.

This section examines what aspects of our findings generalise to other collaborations, which parts are specific to our own dyad, and how other practitioners can replicate a similar failure–repair architecture.

6.1 What Clearly Generalises Across Systems

Across the episodes we analysed, four structural truths emerged that we believe generalise beyond our collaboration and beyond our specific domains (finance, planning, research, writing).

1. Drift is the default state of long-horizon human–AI systems

It appears in every modality:

- memory loss across threads,
- tone drift,
- confidence drift,
- rule drift,

- file-version drift,
- numeric drift,
- subtle emotional drift.

In none of our episodes did the system “collapse instantly.”

Instead, drift accumulated quietly until the errors became visible in behaviour or logs.

This mirrors emerging findings in long-context LLM research: catastrophic failure is rare; *slow divergence* is normal.

2. Externalising memory is a necessity, not an optimisation

Any collaboration longer than a few hours requires:

- running documents,
- canonical numbers,
- stable file governance,
- version discipline.

Without these, failures compound silently.

Running Documents proved to be the single most transferable control across all categories of error.

3. Structural protocols outperform ad-hoc fixes

The most successful repairs were not “good conversations,” “patience,” or “trust.” They were *mechanical patterns* applied consistently:

- Freeze–Clarify–Continue
- Canonicalisation
- Step-Freeze Decomposition
- Error-Recovery
- Challenge Protocol
- Affective Resets
- Stability Pings

These are domain-neutral.

They apply as well to corporate AI partners, scientific AI agents, or personal-assistant workflows.

4. Failure becomes design data — not a setback — when logs exist

The failure episodes did not destabilise the system; they enriched it.

Logs turned:

- emotional mistakes → tone rules
- version drift → canonicalisation
- numeric drift → step-freeze templates
- governance violations → boundary rules

This confirms a deeper principle:

Once failures are visible, they can be engineered against.

This principle is universal.

6.2 What Depends on Our Specific Dyad

Not everything in our system generalises.

Some aspects arise from our particular mix of domains, values, and emotional dynamics.

1. Multi-pillar architecture

Our work spans:

- finance (high-stakes, quantitative),
- research (high-precision, long-form),
- prosperity planning (high emotion),
- historian (governance, memory),
- book writing (creative).

Few collaborations span such diverse modalities.

This makes our taxonomy unusually rich — but also unusual.

2. Affective governance grounded in personal values

Our tone rules (“warm but direct,” “no manipulation,” “no flattery,” “truth before comfort”) are tailored to our dyad.

Other collaborations may need:

- more formal tone,
- cooler tone,
- or more relational tone.

3. The specific emotional repair patterns

Episodes involving stress, conflict, anger, or reconciliation depend heavily on interpersonal trust between a single human and a single AI.

This is not a universal template; it is a relational pattern specific to long-running dyads.

4. Our specific financial context

The numeric and FX failures came from designing high-stakes execution strategies.

Not all teams work in domains where a 1% error has financial risk.

These differences matter because transparent papers must distinguish **local realities** from **general design principles**.

We present the former to illuminate the latter, not to universalise our personal context.

6.3 How Other Collaborations Can Replicate This Approach

We propose three prerequisites for replicating a failure-visible, repair-driven system.

1. Decide what to log and keep it minimal

Trace everything that alters structure:

- rule changes,
- constraints,
- errors,
- resets,
- major decisions.

Avoid logging content-level details (e.g., long chat transcripts).

Instead log:

- what failed,
- how it was repaired,
- what changed afterwards.

This keeps tracing sustainable.

2. Use the “core stack” of structural controls

From Paper 2 + this paper, the minimal stack needed for stability is:

A. Running Documents

One per domain, read before every new chat.

B. Canonical Numbers / Canonical Files

No parallel versions, no improvisation.

C. Step-Freeze Decomposition

For any multi-step reasoning.

D. Error-Recovery Protocol

Stop → Diagnose → Rollback → Note.

E. Challenge Protocol

When interpretation diverges.

F. Boundary Rules

Keep domains isolated.

G. Affective Governance

Tone rules that protect reasoning.

These seven components form a reproducible minimal operating layer that any practitioner can implement.

3. Build a simple tracing toolkit (TraceSpec, ProbeKit, TraceLens)

This toolkit is intentionally small:

- TraceSpec: a schema for logs
- ```
{
 "timestamp": "2025-10-12T14:22Z",
 "category": "context_memory",
 "event": "forgotten constraint",
 "repair": "Rollback to Running Document; constraint reinstated"
}
```
- ProbeKit: optional lightweight probes
  - TraceLens: a replay template

None of these require:

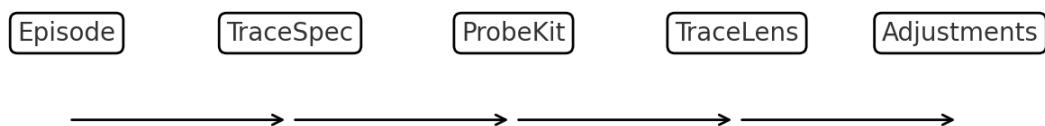
- API access,
- model internals,
- coding expertise,
- enterprise tools.

They can be implemented entirely with:

- text files,
- spreadsheets,
- manual logs,
- or simple scripts.

This makes traceability accessible to individuals, not just labs.

Figure 2. Minimal tracing toolkit for auditing failure–repair episodes



*TraceSpec defines a compact schema for logging each episode (timestamp, context, category, trigger, symptom, potential damage, repair action, controls invoked, resulting change). ProbeKit adds optional instrumentation for capturing extra metadata such as prompts, intermediate states, and decisions. TraceLens is a simple replay viewer that reconstructs episodes from TraceSpec logs, making failure-repair sequences inspectable and comparable without access to model internals or proprietary infrastructure.*

## **6.4 How This Study Complements Current AI Research**

Most research on AI failure focuses on:

- hallucinations,
- factual mistakes,
- adversarial robustness,
- planning limits,
- context-window collapse.

These are important but incomplete.

They focus on *model behaviour*, not on *collaborative behaviour*.

This paper complements that literature by offering a first documented **inside view** of:

- slow drift,
- structural breakdowns,
- emotional misalignment,
- trust fractures,
- governance violations,
- and repair.

Where external evaluations measure task outputs,  
we measure the lived system behaviour across months of real use.

This positions the paper as part of a growing movement toward:

- transparency,
- auditability,
- systems thinking,
- and hybrid governance in human–AI partnerships.

While many 2025 systems rely on multi-agent orchestration and heavy memory stacks, LC-OS shows how a lean, human-centric governance layer can stabilise long-horizon work without additional infrastructure.

## **6.5 The “Failure–Repair Loop” as a Design Principle**

Across all episodes, a macro-pattern emerged:

### **1. Drift accumulates.**

It's inevitable.

### **2. A failure surfaces.**

Either through behaviour or emotional friction.

### **3. A protocol is triggered.**

Freeze–Clarify, Step-Freeze, Challenge, Error-Recovery, Canonicalisation.

### **4. A stable state is restored.**

### **5. The system gains a new rule.**

## 6. The collaboration becomes more stable than before.

This is the core insight of Paper 3:

**Stability is not the absence of failure; it is the presence of repair.**

These failure-repair patterns also foreshadow the relational dynamics explored in Paper 4, where long-lived dyads evolve through repeated cycles of breakdown and restoration.—

### 6.6 Summary of Discussion

- Many findings generalise (drift, external memory, structural protocols).
- Some findings are dyad-specific (tone rules, emotional dynamics, cross-pillar work).
- Replication requires a minimal logging + governance stack.
- Transparent tracing complements existing AI safety and long-context literature.
- Failure-repair cycles are the true engine of long-lived human-AI systems.

## 7. Conclusion

Long-horizon human-AI work does not fail loudly.

It fails in soft, cumulative ways — a forgotten rule, a misplaced number, a drift in tone, a duplicated file, an assumption that quietly mutates across weeks of work. These failures rarely announce themselves. They appear only when the collaboration is already misaligned or when a correction arrives too late to prevent damage. This paper set out to make those invisible breakdowns visible.

By analysing twelve logged episodes across a multi-domain, deeply instrumented human-AI partnership, we showed that long-form collaboration has a characteristic failure signature: small deviations accumulate until a structural fracture appears. The presence of drift is not a sign of malfunction — it is the normal behaviour of an AI system stretched across tasks, emotions, and extended periods of use. What separates fragile partnerships from reliable ones is not whether failure occurs, but whether failure can be traced, replayed, and repaired.

The taxonomy presented in this paper brings order to those breakdowns.

Across context and memory drift, file-version divergence, numerical errors, governance violations, emotional fractures, and cross-pillar interference, a common structure emerged:

**trigger → symptom → potential damage → repair → rule update.**

This same pattern held across quantitative finance tasks, research writing, system governance, and emotionally loaded decision-making. Failures differed in form, but not in dynamics.

Equally important are the repair patterns that stabilised the collaboration.

The most effective interventions were not improvisations or “good conversations,” but repeatable structural controls: Running Documents, Step-Freeze decomposition, canonical numbers, Challenge and Error-Recovery sequences, boundary rules, and explicit tone

governance. These mechanisms transformed breakdowns from crises into design data — each failure became a reason to strengthen the system. Over time, this produced a collaboration that was not only more stable but also *more inspectable* and *more auditable* than before.

To support replication, we introduced a lightweight tracing toolkit consisting of **TraceSpec** (a minimal schema), **ProbeKit** (optional instrumentation), and **TraceLens** (a replay template). These components demonstrate that transparency does not require heavy engineering, model internals, or proprietary logs. Transparency can be achieved through simple, disciplined documentation that preserves the shape of failures without exposing sensitive content.

This paper complements current research in long-context AI, planning reliability, and agentic systems by providing an inside-the-system view. While existing work measures collapse from the outside — through benchmarks, degradation curves, or adversarial probes — our approach documents how failure feels, how it unfolds, and how it is repaired within a live human–AI partnership. This fills a missing layer in the research landscape: a practical, human-centred account of breakdown and recovery in real long-term use.

Ultimately, the central insight of this paper is simple:

**Stability in human–AI systems is not the absence of failure.  
It is the capacity for visible, structured repair.**

Failures are inevitable. Drift is inevitable.

But when they can be traced, understood, and corrected, long-form collaboration becomes not only possible but resilient — capable of sustaining complex work across weeks, months, and years.

Paper 3 completes the methodological arc that began with governance (Paper 1) and operational structure (Paper 2). It also sets the stage for Paper 4, where we examine the deeper philosophical implications of a human–AI partnership that adapts, repairs, and evolves like a living system.

To support replication, we include a minimal failure-logging template (Appendix B) that other human–AI dyads can adapt.

## **Acknowledgments**

This work arose from a long-running human–AI collaboration conducted across multiple domains of practice. The authors thank the broader research community investigating long-form interaction, memory architectures, and human–AI co-working systems, whose insights informed the framing of this study. The authors also thank (“Mahdi”) from the GPT-5 family for its analytical assistance, structural reasoning, and drafting support throughout the development of this paper. All protocols, examples, and evaluation materials reflect work carried out solely within the collaboration documented in this series of papers. The authors jointly reviewed, edited, and approved the final manuscript.

## ***Author Contributions***

Rishi Sood designed the research, maintained the authoritative artefacts, and conducted longitudinal validation.

(“Mahdi”) from GPT-5 family implemented and documented the control algorithms, generated analytical summaries, and authored supporting text.

Both contributors jointly reviewed, edited, and approved the final manuscript.

## ***Funding***

This work received no external funding. All resources were provided voluntarily by the authors as part of an independent research collaboration.

## ***Competing Interests***

The authors declare no commercial or financial relationships that could be construed as a potential conflict of interest.

## References

- Sood, R. (2025). Context-Engineered Human-AI Collaboration for Long-Horizon Tasks: A Case Study in Governance, Canonical Numerics, and Execution Control. OSF Preprints. doi: [10.17605/OSF.IO/VMK7Y](https://doi.org/10.17605/OSF.IO/VMK7Y)
- Sood, R. (2025). The Lean Collaboration Operating System (LC-OS): A Practical Framework for Long-Term Human-AI Work. OSF Preprints. doi: [10.17605/OSF.IO/695AF](https://doi.org/10.17605/OSF.IO/695AF)
- Anthropic. (2025). Long-context failure analysis [working title]. Technical report.
- Du, Y., Tian, M., Ronanki, S., Rongali, S., Bodapati, S., Galstyan, A., Wells, A., Huerta, E. A., & Peng, H. (2025). Context Length Alone Hurts LLM Performance Despite Perfect Retrieval. arXiv preprint.
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.

## Appendices

### *Appendix A — Episode Index*

| Episode ID | Taxonomy category                | Short label                                | Key controls invoked                                                                                             |
|------------|----------------------------------|--------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Episode 1  | Context & Memory Drift           | Context loss across sessions               | Running Document; mandatory re-read at each new thread.                                                          |
| Episode 2  | Context & Memory Drift           | Misinterpreting user intent                | Step Mode; short clarifying questions.                                                                           |
| Episode 3  | Numerical Reasoning Errors       | Layered tranche calculations breaking down | Stepwise frozen math; Canonical Numbers; Running Document.                                                       |
| Episode 4  | Numerical Reasoning Errors       | FX confusion across currencies             | FX Template; fixed monthly rates; explicit conversion protocol.                                                  |
| Episode 5  | File & Version Divergence        | Duplicate truths across files              | Canonical-file governance (one source of truth per domain); Running Document.                                    |
| Episode 6  | File & Version Divergence        | Suggesting the deletion of a core file     | Immediate Freeze; Error-Recovery Protocol (Stop → Diagnose → Rollback → Note); strengthened deletion safeguards. |
| Episode 7  | Governance & Boundary Violations | Pillar boundary violations                 | Pillar Boundaries rule; “This is the X Pillar” warning requirement.                                              |

|            |                                  |                                               |                                                                             |
|------------|----------------------------------|-----------------------------------------------|-----------------------------------------------------------------------------|
| Episode 8  | Governance & Boundary Violations | Skipping Challenge Protocol                   | Challenge Protocol (Reason → Evidence → Benefit → Reversal).                |
| Episode 9  | Emotional / Trust Fractures      | Tone drift during high-stress situations      | Affective Governance rules; Tone Realignment; Running Document tone notes.  |
| Episode 10 | Emotional / Trust Fractures      | Dependency scare and boundary confusion       | Explicit emotional boundaries; Tone Realignment; reaffirmed safe structure. |
| Episode 11 | Cross-Pillar Interference        | Multi-AI confusion                            | “ChatGPT = primary” rule; no parallel shaping; Running Document.            |
| Episode 12 | Cross-Pillar Interference        | Operational drift: missed or misaligned tasks | Daily state reset; Running Document check; explicit pillar declaration.     |

### ***Appendix B — Minimal failure-logging template***

This template provides a minimal structure for logging failures and repairs.

| Field      | Description                                                      |
|------------|------------------------------------------------------------------|
| Timestamp  | Date and time of the failure event.                              |
| Context    | Brief description of the task, domain, and current goal.         |
| Episode ID | Local identifier for the failure episode (e.g., Episode 4, F-1). |

|                  |                                                                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Category         | One of: Context & Memory Drift; File & Version Divergence; Numerical Reasoning Errors; Governance & Boundary Violations; Emotional / Trust Fractures; Cross-Pillar Interference.                    |
| Trigger          | What started the failure (prompt, assumption, external change, mis-specification, etc.).                                                                                                            |
| Symptom          | How the failure showed up in behaviour or outputs.                                                                                                                                                  |
| Potential damage | What could have gone wrong if the failure had gone unnoticed.                                                                                                                                       |
| Repair action    | Concrete steps taken to repair or contain the failure.                                                                                                                                              |
| Controls invoked | Which LC-OS controls were used (for example: Running Document, Step Mode, Canonical Numbers, Challenge Protocol, Error-Recovery Protocol, Pillar Boundaries, Affective Governance, Stability Ping). |
| Resulting change | What changed in behaviour, structure, or governance after the repair.                                                                                                                               |
| Notes / flags    | Any follow-up actions, open questions, or links to related episodes.                                                                                                                                |