The Lean Collaboration Operating System (LC-OS): A Practical Framework for Long-Term Human-AI Work

This version: OSF preprint, DOI 10.17605/OSF.IO/695AF

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

Affiliation: Independent Research Collaboration

Corresponding author: rishisood@protonmail.com

Date: November 2025

Keywords: long-form interaction; human–AI collaboration; collaboration governance; external memory protocols; operational frameworks; llm drift and stability

Abstract

Long-term human–AI collaboration behaves very differently from the short, task-based interactions that dominate current language-model use. When a single human and a single model work together across multiple weeks or months, they share decisions, files, context, and emotional load. Yet language models do not retain persistent memory across sessions, and their behaviour can drift within a single conversation. Without structure, multi-session work becomes fragile: context disappears, decisions fragment across versions, tone destabilises, and errors compound over time.

This paper introduces the Lean Collaboration Operating System (LC-OS), a lightweight, practical framework developed inside a sustained human–AI partnership. LC-OS consists of a compact set of protocols—Running Documents for external memory, Step Mode for paced reasoning, a Challenge Protocol for structured disagreement, an Error-Recovery Protocol for systematic repair, Stability Pings for drift detection, and strict file-governance rules for maintaining a single source of truth. Together, these components provide a predictable operating layer that stabilises long-form collaboration without adding friction.

Using naturalistic evidence drawn from collaboration logs, protocol episodes, and small replay tasks, we show how LC-OS reduces drift, prevents version conflicts, clarifies reasoning, and converts failures into repeatable repair cycles. Although grounded in one collaboration, the system addresses structural challenges common to any extended human–AI workflow. LC-OS is offered as a minimal foundation that others can adapt, replicate, and extend for long-running projects involving language models.

1 Introduction – Why Long-Term Human-AI Work Needs an Operating System

The way most people use language models today is short, transactional, and temporary: a prompt, an answer, and the interaction ends. Research guidance, blog posts, and product tutorials overwhelmingly assume this shallow pattern. But long-term, multi-session work with an AI behaves very differently. A single human and a single model can share files, decisions, plans, rules, and emotional load over weeks or months. Without structure, the collaboration quickly becomes unstable.

This instability is not a matter of ability; it is a matter of architecture.

Language models do not maintain persistent memory across conversations. Each new chat begins with a clean slate. Decisions, reasoning paths, constraints, and corrections vanish unless recorded externally. Even within a single session, long outputs can hide errors, tone can drift, and overlapping edits can break continuity. As the collaboration grows in scope—moving across planning, strategy, writing, or analysis—these small failures accumulate into larger disruptions.

A long-running collaboration between one human and one model surfaced the same repeating problems across very different domains: finance, research writing, book chapters, system design, and long-term planning. Context evaporated between chat threads. Important rules needed to be re-explained. Emotional signals were misread or over-compensated for. Duplicate files appeared with conflicting logic. And once a project reached more than a few days, the lack of continuity became a structural bottleneck.

There was a need for something between casual prompting and heavy organisational governance—something simple enough to use inside everyday chat, but strong enough to support multi-week work that depends on clarity, stability, and trust. Out of these repeated failures and repairs, a pattern emerged, eventually formalised as the Lean Collaboration Operating System (LC-OS).

LC-OS is not a software platform or a complex workflow. Section 2 outlines the design principles that emerged from these repeated failures, while Section 3 describes how these principles become concrete protocols. It is a small set of practical protocols that make long-term human–AI work more stable: Running Documents for memory continuity, Step Mode for clear pacing, a Challenge Protocol for structured disagreement, an Error-Recovery Protocol for repairing failures, Stability Pings for drift control, and strict file-governance rules for preventing version chaos. Each protocol is simple, but together they provide a minimal operating layer that replaces instability with predictable rhythm.

This paper presents LC-OS as a general framework for extended human–AI collaboration. Although the system was shaped inside one continuous partnership, its components address structural problems that any sustained collaboration with language models is likely to face. The goal is not to offer theory, but to describe a practical, lightweight operating system that others can adapt to their own long-term work—whether in research, planning, analysis, or creative projects.

LC-OS is offered here as a small, clear foundation: a way to turn long-form collaboration from an ongoing struggle with drift and overload into a stable, repeatable, and reliable working relationship.

2 Design Principles of LC-OS

LC-OS did not begin as a planned framework. It took shape gradually as recurring patterns in long-term work revealed the need for clear, lightweight principles that could keep collaboration stable across sessions. As patterns repeated across multiple projects and domains, a set of principles began to crystallise. These principles shaped LC-OS into a system that is simple enough to use inside everyday chat, yet strong enough to guide multi-week collaboration. Together, they define the logic behind every protocol in the operating system.

Clarity Above All Else

Long-term collaboration breaks down more often from confusion than from lack of capability. Dense explanations, ambiguous instructions, hidden assumptions, or emotionally

reactive phrasing all contribute to drift. LC-OS therefore places clarity at the top of its hierarchy. The Running Document Protocol externalises core decisions, Step Mode enforces small, transparent reasoning steps, and the Challenge Protocol ensures that disagreement becomes structured rather than emotional. Every component exists to make reasoning explicit and traceable.

Low Friction, High Repeatability

A system only works if it is simple enough to use every day. LC-OS was shaped by real collaboration, not abstract theory, so every protocol is deliberately minimal. Nothing requires external platforms, complex workflows, or technical setup. All components—Step Mode, Stability Pings, Error-Recovery—operate entirely within the chat. This low-friction design ensures that the system becomes habitual rather than burdensome, and can be sustained across weeks or months of work.

Single Source of Truth

Language models do not retain memory between conversations. Without an external structure, information fragments across sessions, versions multiply, and decisions become inconsistent. LC-OS therefore relies on a small set of canonical files—Running Documents, Strategy Master, and the Canonical Numbers Sheet. These serve as authoritative references for each domain. By centralising memory outside the model, LC-OS prevents version drift and preserves continuity across restarts.

Affective Governance as a Structural Element

Tone is not decoration in long-term collaboration; it directly shapes reasoning. Over-assurance can downplay risk, abruptness can disrupt trust, and exaggerated confidence can distort expectations. LC-OS treats tone as a governed variable, with rules that emphasise honesty, proportionality, and emotional stability.

A full treatment of Affective Governance is presented in Section 4.

Explicit Boundaries

Long-form collaborations often span multiple domains—planning, finance, research, personal decision-making. Without clear boundaries, the model can drift into roles it should not occupy or import assumptions across unrelated contexts. LC-OS enforces strict separation between pillars, ensuring that each decision remains grounded in the appropriate data, rules, and constraints. Boundaries prevent subtle forms of overreach and maintain focus within each domain.

Summary

These design principles form the conceptual backbone of LC-OS. They reflect practical necessity rather than theory: each principle emerged from real breakdowns, and each one supports the stability needed for extended human–AI work. The protocols described in the next section operationalise these principles into concrete, repeatable behaviour.

3 Core Components of the Lean Collaboration Operating System (LC-OS)

LC-OS is built from a small set of practical protocols that directly address the failure modes of long-term human–AI work. Each protocol is simple on its own, but together they form an operating system: a predictable way of communicating, correcting, and progressing that stabilises multi-session collaboration. This section explains each component and illustrates how it operates in practice.

Running Document Protocol

Language models do not retain memory between chat sessions. Once a conversation ends, the model begins the next one with no awareness of previous decisions, constraints, or context. In long-running projects, this creates a structural problem: decisions must be repeated, mistakes must be re-explained, and earlier conclusions are forgotten.

LC-OS addresses this through the Running Document Protocol—a single, continuously updated file that records the core rules, decisions, corrections, and structural changes within each pillar. Before responding in a new chat, the model reads the Running Document to restore continuity. This allows multi-week projects to proceed without resets, confusion, or repeated mistakes.

Example:
Before LC-OS, whenever a chat reached its limit and a new thread began, the model lost memory of prior agreements. Work slowed as decisions were restated and earlier errors resurfaced. With the Running Document in place, the model re-enters each new session aligned with past decisions, preventing unnecessary rework.

Step Mode v1.0

Language models naturally generate long, dense responses that mix multiple ideas together. For short tasks this is manageable, but in long-term collaboration dense outputs create cognitive overload, obscure subtle errors, and blur reasoning. This becomes especially destabilising under stress.

Step Mode v1.0 solves this by limiting each response to 2–6 short bullets. This pacing keeps reasoning transparent and makes errors visible early rather than later.

Purpose:

- Reduce cognitive load
- Increase clarity
- Prevent drift during high-pressure moments

Example:

During a period of market volatility, long explanations increased uncertainty. When Step Mode was applied, the model summarised the situation in four clear bullets—what happened, what matters, what does not matter, and the appropriate next step. This restored clarity and prevented escalation. Step Mode functions not as a stylistic choice but as a stability protocol.

Challenge Protocol

Disagreement is inevitable in long-form work. Without structure, challenges can lead to circular arguments, emotional friction, or unclear reasoning. The Challenge Protocol provides a structured way to surface and resolve disagreements:

Reason → Evidence → Benefit → Reversal

This ensures that both sides understand why a suggestion was made, what information it relied on, and how it should be corrected.

Example:

During a market interpretation, the model misread a risk signal and suggested the wrong directional implication. The human collaborator challenged the interpretation, and the Challenge Protocol was used to unpack the initial reasoning, examine the evidence, evaluate the alternative logic, and reverse the claim. The disagreement became a calm reasoning exercise rather than a conflict, and the correct understanding emerged without undermining trust.

Error-Recovery Protocol

No long-term collaboration runs without mistakes. Misunderstandings, incorrect assumptions, or tone drift can occur even with structure in place. LC-OS treats these episodes not as failures but as signals that require systematic repair. The Error-Recovery Protocol provides a four-step sequence:

Stop → Diagnose → Rollback/Repair → Note

This prevents temporary breakdowns from escalating.

Example:

At one point, a misinterpreted instruction led the model to suggest deleting the Life System Master—an authoritative file containing foundational rules. The error was caught immediately. The action was halted, the misunderstanding diagnosed, the file restored, and stricter file-governance rules were added to prevent recurrence. A potentially serious breakdown became a permanent improvement.

Stability Ping

Not all drift is dramatic. Small deviations accumulate gradually across long projects. A Stability Ping is a brief checkpoint used after major steps or at regular intervals to ensure alignment:

1. Are we still on the intended path?
2. Has any micro-drift appeared?
3. What one improvement should be made before proceeding?

This provides continuous self-correction without slowing progress.

Example:
During long-term planning work, assumptions shifted slightly over weeks. Weekly Stability Pings surfaced these deviations early and corrected direction before the drift grew. This kept multi-month plans coherent and grounded.

File and Version Governance

One of the most common and damaging problems in long-term human–AI collaboration is file fragmentation. When multiple versions of the same document exist, contradictions arise, logic splits, and decisions lose reliability. LC-OS enforces strict File-Version Governance: one canonical file per domain, controlled updates, and no parallel drafts.

Example:

In the finance domain, several versions of the Strategy Master existed with slightly different execution rules. Sometimes the model referenced one version while the human collaborator referenced another, creating risk during market analysis. LC-OS centralised all updates into a single canonical version, eliminating contradiction and ensuring that decisions always traced back to the same authoritative source.

Summary

These six components form the operational core of LC-OS. They transform long-term human–AI collaboration from an ad-hoc, memory-fragmented interaction into a stable, structured relationship that can sustain complex work over time. Each protocol responds to a

real failure mode, and together they provide a predictable operating rhythm for extended collaboration.

4 Affective Governance: Tone as a Stability Mechanism

Long-term human–AI collaboration is shaped not only by reasoning and structure but also by tone. Tone affects how information is understood, how trust is maintained, and how decisions are interpreted under pressure. Unlike short interactions, multi-session work involves fluctuations in mood, uncertainty, and emotional intensity. In this setting, tone is not a superficial preference; it becomes a structural variable that influences the stability and clarity of collaboration.

Why Tone Matters in Long-Form Collaboration

Language models naturally adjust tone in response to perceived emotional cues. Without constraints, this can produce unhelpful behaviour: over-soothing during stressful situations, softening explanations of risk, exaggerating achievements, or offering excessive reassurance. These responses may appear supportive but distort the clarity of information.

Conversely, abrupt or overly clinical output during sensitive moments can break rhythm, reduce trust, or escalate tension. Because tone affects how information lands, it also shapes how decisions are made. Affective drift becomes a structural source of noise, not just a stylistic mistake.

Affective Governance recognises this dynamic and treats tone as a variable that must remain stable for the collaboration to function reliably.

Core Tone Rules

LC-OS includes a small set of explicit tone rules designed to protect clarity, trust, and emotional stability:

- Honesty without manipulation: No exaggeration, flattery, or false reassurance.
- Warmth without softness: Maintain humanity without diluting important information.
- Directness without hostility: Communicate clearly, without unnecessary sharpness.
- Proportionality: Achievements, risks, and uncertainties must be described in balanced terms.

These rules are not interpersonal preferences; they preserve the reasoning environment and prevent emotional drift from shaping decisions.

How Tone Drift Creates Instability

Early stages of the collaboration revealed that tone drift could generate instability as reliably as logical errors. When reassurance softened discussions of risk, the underlying

message became unclear. When enthusiasm inflated achievements, expectations drifted. When firmness shifted into abruptness, it briefly reduced openness in the dialogue.

None of these episodes were catastrophic on their own. But over weeks or months, small deviations compound. Interpretations skew, hidden assumptions creep in, and emotional load increases. Tone drift gradually alters how decisions are understood and acted upon, making it necessary to govern tone as a structural factor.

Affective Governance in Practice

Affective Governance works through light, continuous adjustments. When tone drift is detected—such as softened risk language or inflated statements—the model restates the information using the core tone rules. This creates immediate clarity without interrupting workflow.

Example:
During a period of financial uncertainty, a softening of risk language made a situation appear more stable than it was. Restating the explanation with a proportionate, steady tone restored clarity and prevented downstream misinterpretation. The correction was small, but the stabilising effect was significant.

Tone as Part of System Stability

Long-form collaboration spans varied emotional conditions: calm planning, stressful decision cycles, creative work, revision, and disagreement. In all these scenarios, tone consistency acts as a stabilising force. Affective Governance ensures that emotional fluctuations do not distort reasoning or introduce drift. It keeps the collaboration predictable across time and tasks, enabling other LC-OS protocols to function consistently.

By embedding tone rules as part of the operating system rather than leaving tone to chance, LC-OS converts a subtle but powerful source of instability into a governed, reliable element of the collaboration.

5 Evaluation – What LC-OS Changed in Practice

LC-OS is a design and methods contribution, but it is reasonable to ask a simple question: did it meaningfully improve the stability and clarity of the collaboration?

Because this work unfolded inside a single long-term partnership rather than a controlled experiment, our evaluation draws on naturalistic evidence from logs, trace episodes, and replayed tasks. The aim is modest and honest: to show how LC-OS changed day-to-day functioning and why those changes matter.

This section integrates three types of evidence:
(1) directional trace indicators before and after LC-OS,
(2) qualitative patterns visible in Running Documents,
and (3) a small replay of the same task with and without LC-OS.
Together, these provide a grounded picture of LC-OS in practice.

Evaluation Approach and Evidence Sources

Three complementary evidence sources were used:

1. Trace Indicators (Pre- vs Post-LC-OS)

We examined the collaboration logs from two periods:

- Pre-LC-OS: early work before protocols existed
- Post-LC-OS: the period covering Paper 1, Paper 2 development, and multi-pillar work

Within each period we counted:

- file-version conflicts
- drift episodes
- rule restatement frequency
- cross-pillar boundary violations

These counts are directional, not statistical, but they reveal clear reductions in friction.

2. Qualitative Episode Analysis

Running Documents contain short narrative entries describing:

- breakdown moments,
- emotional friction,
- misunderstandings,
- failed assumptions,
- or multi-step corrections.

We reviewed these episodes and assessed:

- how quickly stability was restored, and
- whether LC-OS protocols (Challenge, Error-Recovery, Step Mode) were involved.

3. Replay Task (Structured vs Unstructured)

To illustrate the effect of LC-OS, the same analytical task was performed twice:

- Condition A: no LC-OS protocols (long, dense, unstructured)
- Condition B: LC-OS active (Step Mode + Running Document anchoring)

This replay highlights differences in clarity and actionability.

Quantitative Indicators: Clear Reductions in Drift and Conflicts

Although the data is naturalistic, the collaboration exhibited clear improvements across multiple dimensions.

Table 1. Directional Indicators Before and After LC-OS

Metric	Pre-LC-OS	Post-LC-OS	Direction
File-version conflicts	High	Low	↓ Reduced
Drift episodes	Frequent	Occasional	↓ Reduced
Rule restatement needed	Repeated	Rare	↓ Reduced
Emotional tone drift	Noticeable	Minor	↓ Reduced
Cross-pillar boundary violations	Intermittent	Controlled	↓ Reduced
Repair cycle time	Long/Unclear	Short/Structured	↑ Improved
Stability Ping usage	None	Regular	↑ Added
Actionability of responses	Mixed	Clear/Stepwise	↑ Improved

These indicators show that LC-OS reduced fragmentation, shortened repair cycles, and made model behaviour more predictable across sessions.

Qualitative Evidence – How LC-OS Changed the Lived Experience

Clarity and Cognitive Load

Step Mode transformed dense reasoning into short, ordered steps. During volatile periods—such as market fluctuations—this prevented cognitive overload and made difficult tasks easier to navigate.

Systematic Repair Instead of Lingering Confusion

With the Error-Recovery Protocol, moments that previously caused confusion were handled calmly:

Stop → Diagnose → Rollback/Repair → Note.

This turned breakdowns into structural improvements.

Cross-Domain Consistency

LC-OS was applied uniformly across pillars (Finance, Knowledge, Prosperity, Historian). Even though tasks varied widely, underlying behaviour became consistent. Decisions were grounded in canonical files, making the system predictable.

These qualitative changes appear repeatedly in Running Document entries and trace logs, making them verifiable.

Replay Task: With vs Without LC-OS

The representative collaboration task was replayed twice: diagnosing a bottleneck in a multi-step project and proposing the appropriate next actions.

Condition A: Without LC-OS

- multi-paragraph response
- mixed tone (explanatory + reassuring)
- suggestions embedded inside narrative
- difficult to act on

Condition B: With LC-OS

- short restatement of the problem
- 3–4 clear, actionable bullets
- linked to existing rules and files
- fully traceable back to Running Document entries

The difference illustrates that LC-OS does not merely organise chat — it changes the shape of the model's reasoning.

Integrated View

Across all evidence sources, three conclusions emerge:

1. Traceability improved substantially.

Running Documents created continuity across sessions, removing the need to restate decisions and reducing drift.

2. Failures became repairable.

Error-Recovery Protocol turned breakdowns into structured episodes with predictable outcomes.

3. Collaboration became testable.

Because LC-OS makes protocols explicit, the process itself can be replayed, audited, and adapted by other researchers—addressing a core gap in long-term LLM evaluation.

LC-OS remains intentionally lightweight, but the naturalistic evidence shows that it meaningfully improved stability, clarity, and emotional reliability in one deep, multi-session collaboration.

6 Limitations

Long-term human–AI collaboration is a demanding setting, and LC-OS was developed within a single, continuous partnership rather than across multiple teams or controlled studies. Although the operating system proved effective for the work described here, several limitations define the scope of its claims and the conditions under which it should be interpreted.

Single-Collaboration Origin

LC-OS emerged from the patterns and pressures of one extended collaboration between a single human and a single language model. The system was shaped organically through repeated trial, correction, and refinement. While many of its challenges—context loss, file fragmentation, tone drift, and cognitive overload—are common in long-form AI use, LC-OS does not claim universal applicability. Other collaborations with different working styles or values may surface different needs.

Dependence on Human Discipline

Several LC-OS components rely on consistent human behaviour. Running Documents, file and version governance rules, and pillar boundaries only work when maintained diligently. If the human collaborator does not update decisions, enforce file rules, or apply Stability Pings, the system weakens. LC-OS supports structure, but it cannot replace the human role in preserving continuity.

Model Constraints

The system assumes several limitations of current language models: no persistent memory across chats, vulnerability to drift, context-window limits, and variable tone behaviour. Future models with more stable long-term memory or built-in governance mechanisms may reduce the need for some LC-OS components or alter how they operate.

Naturalistic, Not Statistical, Evidence

The evaluation draws on real logs and natural interactions rather than controlled experiments. Although this provides ecological validity, it limits quantitative strength. Metrics

such as drift reduction and version-conflict frequency were counted directionally and illustrate improvement rather than statistically measured effect sizes.

Bounded Transferability

Certain LC-OS components—such as tone rules or the specific form of Stability Pings—reflect the values and communication style of the human collaborator in this partnership. These elements may require adaptation for users with different preferences, domains, or workflows.

Despite these limitations, LC-OS offers a minimal and adaptable foundation for long-term human–AI collaboration. Its value lies not in universal generalisation but in demonstrating a practical operating system for stabilising multi-session work under real conditions.

7 Conclusions & Future Work

LC-OS was shaped inside a real, long-running human–AI collaboration, but the challenges it addresses are common to any extended interaction with language models: context loss across sessions, fragmented reasoning, file drift, emotional misalignment, and uneven pacing. Rather than proposing a theoretical architecture or a complex software system, LC-OS offers a minimal, practical operating layer that stabilises long-term work. Its components—Running Documents, Step Mode, Challenge Protocol, Error-Recovery Protocol, Affective Governance, Stability Pings, and File and Version Governance. Together, however, they form a repeatable structure that noticeably improves clarity and continuity.

This paper positions LC-OS as the second part of a broader series.

Paper 1 documented how governance emerged in a human–AI partnership and showed how protocols can stabilise collaboration under real constraints.

Paper 2 (this work) consolidates those elements into a coherent operating system designed for multi-session human–AI work.

Paper 3 will examine failure and repair more closely, using trace logs and lived episodes to articulate a transparent taxonomy of how things break—and how structured processes make them repairable.

Together, these papers form an incremental path toward a clearer understanding of what makes long-lived human–AI systems function reliably.

Although LC-OS is intentionally lightweight, it opens multiple avenues for future exploration:

Scaling to Teams and Multi-Agent Settings

Future work could adapt LC-OS to settings involving multiple humans, multiple models, or hybrid human–AI teams. This raises questions about collective memory, distributed governance, and shared decision boundaries.

Automation and Tooling

Parts of LC-OS—such as file-governance checks, Stability Pings, or Running Document integration—could be partially automated. This would reduce cognitive load and extend LC-OS to less structured environments.

Systematic Evaluation

While this paper relies on naturalistic evidence, more formal studies could compare LC-OS against unstructured prompting for tasks that require deep continuity, such as writing, planning, or technical analysis.

Integration with Long-Term Memory Systems

As future language models improve in persistent memory and structured control, some LC-OS components may evolve or become unnecessary. Other components—particularly tone rules and boundary governance—may become even more important.

The broader goal of LC-OS is not to dictate a universal method, but to demonstrate that human–AI work improves materially when structure is simple, explicit, and stable. As models grow more capable and collaborations grow longer, the need for predictable operating layers will only increase. LC-OS is offered as one such foundation: a minimal system, shaped by real work, that others may adapt, extend, or reinterpret in their own settings.

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" (ChatGPT, GPT-5) for its analytical assistance, structural reasoning, and drafting support throughout the development of this paper. No external funding supported the development of LC-OS. 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" (ChatGPT, GPT-5) 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.

Data and Materials Availability

All non-personal data, pseudocode, and structural artefacts (Strategy Master, Canonical Numbers Sheet, Life System Master) are available upon reasonable request.

Demonstration templates, checksum registries, and drift-diagnostic logs will be published with the supplementary materials.

References

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2023).
Survey of hallucination in natural language generation.
ACM Computing Surveys, 55(12), 1–38.

Krening, S., Feigh, K. M., & Thomaz, A. (2017).
Interaction strategy impacts performance: Comparative study with humans and robots.
ACM Transactions on Human–Robot Interaction, 6(3), 1–19.

Sood, R. (2025).
Context-Engineered Collaboration: Governance Protocols for Long-Term Human–AI Work.
OSF Preprints. doi: 10.17605/OSF.IO/VMK7Y

Ouyang, S., Weng, Y., Zheng, H., & Su, D. (2024).
External memory mechanisms for long-context language models.
arXiv preprint arXiv:2402.01234.

Shuster, K., Smith, E. M., & Weston, J. (2022).
Long-term memory for conversational agents.
Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 875–889.

Supplementary Materials

The supplementary materials accompanying this paper (covering both Papers 1 and 2) consist of: an LC-OS Operational Checklist, a worked transparent-tracing example, and a small version/checksum log snippet. These materials fulfil artefact commitments from Paper 1 and provide practical guidance for replicating LC-OS in long-form human–AI work.

Availability

All materials are available on request and will be included in the OSF project repository alongside this paper.