FIELD NOTE · HUMAN-GOVERNED AGENT SYSTEMS

Recursive Learning Without Surrendering Human Judgment

What I've learned about turning memory, correction, and real work into durable improvement.

> A system does not learn because it remembers more. It learns when evidence changes future behavior—and that change remains inspectable, testable, and reversible.

Prepared by Cj TruHeart

July 2026

Discussion brief · v1

01 · THE ESSENTIAL DISTINCTION

Remembering is necessary. It is not yet learning.

The first breakthrough is relieving the human of reconstruction: who said what, what was promised, where a project stands, and what context matters now. The second breakthrough is building a system that improves because of what happened—not merely one that can retrieve it.

01

Memory

Preserves context, decisions, corrections, outcomes, and provenance so the system can reconstruct state.

02

Learning

Turns meaningful evidence into a bounded proposal for a changed rule, workflow, skill, evaluation, or decision path.

03

Authority

Determines which changes may become durable, who approves them, and how they can be reversed.

Verification is the gate between learning and authority. A candidate lesson must survive evidence, replay, or review before it earns permission to alter future behavior.

> “Remember this” is a retrieval instruction. “Change how we operate, prove it works, and preserve the reason” is a learning loop.

The design consequence

A trustworthy recursive system must keep these functions connected without collapsing them. Memory without learning becomes an archive. Learning without verification becomes drift. Verification without human authority can optimize the wrong objective with great efficiency.

Scope: This is not a claim of autonomous self-improvement. It is a practical architecture for human-governed improvement around real workflows.

02 · THE OPERATING LOOP

Turn lived work into inspectable change.

The loop begins close to reality: a correction, a failure, a decision, an unexpected success, or a repeated point of friction.

STEP 1
Capture
Preserve the event, correction, outcome, and source while context is fresh.

→

STEP 2
Classify
Decision, pattern, drift, task, incident, or improvement opportunity?

→

STEP 3
Externalize
Turn the signal into a proposal, issue, checklist, skill, rule, or evaluation.

→

STEP 4
Review
Decide whether to ignore, reference, test, propose, or integrate.

STEP 5
Integrate
Apply the smallest bounded change to the right layer—not everywhere.

→

STEP 6
Verify
Use replay, tests, traces, or a gap-finding pass to challenge the change.

→

STEP 7
Persist
Record what changed, why, who approved it, and how to reverse it.

→

STEP 8 ↻ STEP 1
Calibrate
Use accepted and rejected changes to improve what the loop notices—and captures—next.

The learning check

What appears true or promising? What does it change, if anything? What would falsify it? What is the smallest safe next action? “Learn” does not mean “adopt.”

> The unit of progress is not the conversation. It is the verified behavior change produced by the conversation.

Correct near the mistake. Promote only after proof.

Near-real-time correction is powerful because it preserves the decision context. But a correction should become durable only after the system can explain the failure and survive a replay.

This is the correction subprocess inside the larger operating loop—not a competing version of it.

1. **Locate the error.** Attach feedback to the specific decision, tool call, assumption, or handoff—not only the final output.
2. **Preserve the trace.** Record intent, available context, action, outcome, and the human correction.
3. **Extract the candidate lesson.** State the smallest reusable principle. Separate a one-off preference from a general failure pattern.
4. **Choose the right artifact.** Memory for context; a rule for a boundary; a skill for repeated execution; an eval for measurable behavior.
5. **Test the change.** Replay the original case and at least one adjacent case. Look for unintended rigidity or over-correction.
6. **Promote—or reject.** Integrate only when evidence supports it. Keep rejected proposals as calibration data, not silent failures.

Durable when…

- the source and reason remain traceable.
- the scope and authority are explicit.
- the behavior can be replayed or inspected.
- there is a rollback path.

Dangerous when…

- one correction silently becomes universal doctrine.
- the system edits its own guardrails.
- success is judged by the same process that produced it.
- memory volume is mistaken for wisdom.

04 · ARCHITECTURE

Keep the learning layer independent of the model.

Models will keep changing. A durable system stores authority, state, evidence, and evaluation outside any single model provider.

Human authority	**Intent, values, consent, consequential decisions.** The human decides what the system is for and which changes are allowed to become durable.
Governance	**Permissions, proposal/canon boundaries, privacy, rollback.** These controls must not depend on the model voluntarily obeying them.
Learning harness	**Capture, classification, review, workflow selection, persistence.** This is the mechanism that turns evidence into bounded change.
Verification	**Evals, replays, traces, checks, independent review.** The producer should not be the sole judge of completion or improvement.
Sovereign state	**Files, knowledge, decisions, receipts, task state.** Portable, inspectable, attributable, and recoverable across models and interfaces.
Model cockpit	**Claude, ChatGPT, Gemini, Grok, local models, or future systems.** Choose by task fit; do not let interface convenience become architectural captivity.

What changes often	What should remain stable
Models and pricing	Human authority, data ownership, and permission boundaries
Interfaces and agents	Canonical state, provenance, and durable decision records
Prompts and workflows	Verification criteria, rollback, and escalation paths

> I’m ultimately more attached to the learning architecture than to the substrate.

05 · WORKING PRINCIPLES

The rules I’m carrying forward.

Capture outcomes, not just conversations.

Preserve enough evidence to distinguish success, partial success, failure, and luck.

Build evaluations from painful reality.

Repeated failures and high-value workflows are better teachers than generic benchmarks.

Automate toil. Escalate judgment.

Let agents reconstruct, prepare, monitor, and verify. Keep consequential authority human.

Correct locally before generalizing.

Start with the smallest rule that explains the failure. Widen only when evidence earns it.

Make every durable change legible.

A future human should be able to answer: what changed, why, based on what, and how do we undo it?

Treat rejected changes as learning.

Rejection calibrates the system’s pattern detection and protects against confident drift.

Questions I’d like to compare notes on

1. How do your near-real-time corrections become durable rules?
2. What mechanism distinguishes a one-off preference from a general lesson?
3. How does the system detect, verify, and repair its own mistakes?
4. Which boundaries remain outside the system’s authority to change?
5. What evidence tells you the loop is truly improving—not merely becoming more agreeable?

The goal is not an AI that quietly takes over more judgment. It is a system that returns human attention to the places where judgment, presence, and relationship matter most.

Selected influence: Lee Robinson, “Recursive Model Improvement” (AI Engineer / Cursor), adapted through live use in a local-first, human-governed agent system.