ARKALABS WORKS · DESIGN ESSAY

From Memory to Agency

How an AI agent accumulates experience without letting its past call the shots

THE STARTING POINT

For several months, I have been building Cortex.deck around a simple question: how can an agent avoid rediscovering everything on every new mission?

At first, I mostly wanted to preserve context. I gradually had to distinguish a conversation, a trace, an experience, a learning and a memory. Then a harder problem appeared: as soon as past information can change a future decision, memory receives power.

This essay follows that path from product design. It treats agent memory as a systems problem: what is captured, linked, reconstructed, authorized and ultimately used.

Yesterday, someone told me exactly what I had said three weeks earlier.

I remembered the conversation. I had shared a hunch: something would eventually have an impact. He remembered something stronger: "You said it was going to break everything."

That version sounded exaggerated. He checked his notes. The phrase wasn't there. He just remembered it that way.

The interesting part wasn't who was right. It was watching two people reconstruct entirely different memories from the same conversation. And how much those reconstructions had shaped his decisions over three weeks.

This is the problem I face every day with AI agents.

01

The model is not the agent

When we talk about artificial intelligence, we confuse the model and the agent. They are not the same thing.

A model like GPT or Claude is a reasoning engine. It takes a context, produces a response, and stops. Its parameters don't change because a conversation went well.

The agent is the system built around that engine: tools, permissions, files, rules, memory. The same model can power very different agents. Give it a different context and its behavior changes, even though the engine stays identical.

I've used several models in a single Cortex.deck session. They didn't reason the same way, but I never felt like I was switching collaborators. The engine changed. The system persisted.

That led me to a distinction that shapes everything else: experience can belong to the system, not the model.

02

Why a memory file isn't enough

The first idea looks obvious: save the conversation history. Many agents have a memory file or a full transcript they reread at startup.

Imagine taking notes after every meeting for a year. Two months later, starting a new project, would you reread all your notebooks? Probably not.

Yet that's exactly what we ask agents to do. We inject the full file into the context. Important information gets buried amid outdated decisions and irrelevant details. The token cost is only part of the problem. The noise distorts the reasoning.

I initially thought moving to structured JSON would fix everything. It was a real improvement. But the underlying problem came back fast: content stayed frozen in files, and the agent writing into them could destroy the very material that explained how a conclusion had been reached.

I needed a database. I chose SurrealDB because it runs directly inside Cortex and combines several capabilities in one engine: structured storage, keyword and semantic search, and a native graph model.

03

The detective's case board

A graph is easy to picture. Think of a detective's case board. A witness claims they saw a red car leaving the scene at 10 PM. Detectives identify the owner. A first hypothesis takes shape.

The next day, surveillance footage shows the license plate was misread. Another piece of evidence places the owner miles away.

In a notebook, you'd cross out the old conclusion and write the new one. But part of the investigation would vanish with it. Why was this person suspected? What made the lead look credible? What eventually broke it?

On a case board, nothing needs to be erased. The testimony stays connected to the first hypothesis by a red string labeled "supports." The footage is connected to the same hypothesis by another string labeled "contradicts." Each connection has a date and a status.

That's what Cortex does. When new evidence arrives, the system doesn't silently rewrite the past. It adds to the chain.

04

When a conversation becomes an experience

A conversation starts when you open a chat and ends when you close it. Work rarely follows that boundary.

I found a better mental model in TV shows. Each conversation turn resembles a scene. An early scene introduces the problem. Another sets up the plot. A decision is made, an action follows, and a later scene reveals the result. Watched in isolation, the middle scene loses its meaning if the opening scene has disappeared.

The assembly of these scenes forms what I call an Episode. An Episode connects a starting situation, a goal, the decisions made, the actions taken, and the moment understanding shifted. It doesn't manufacture a story cleaner than the sources. Every element stays linked to the exchanges that produced it.

Multiple Episodes compose a season and build a broader understanding. The next season might reveal that an earlier interpretation was incomplete. The story grows without requiring previous versions to be erased.

05

The cockpit warning light

Imagine a plane. A warning light comes on in the cockpit. The captain has information, but not a conclusion.

Is it an isolated alert? A confirmed failure? A known sensor glitch? The visible signal stays the same. But depending on how it is interpreted, the checklist to consult, the checks to run, and the actions authorized will be completely different.

The checklist doesn't change the information. It constrains the power you give it.

I ran into the same problem with Episodes. A failure can produce several interpretations: "never use X," "X fails when Y is missing," or "the tool was down that day." These statements don't describe the same thing. The first proposes a general rule. The last reports a local observation.

If everything gets filed under "lesson learned," a single incident becomes a blanket prohibition. The agent hits a wall once and starts avoiding the situation everywhere.

That's how an agent becomes superstitious.

06

Keeping a mistake without letting it act

Sooner or later, new evidence contradicts what the system learned.

Deleting the old belief prevents its return, but destroys what made it plausible. Keeping everything with a "do not use" tag preserves the history, but the guarantee is weak. If the text finds its way back into the context, the model can still blend it with the correction.

You have to separate preservation and authorization.

That old checklist from our hypothetical plane was based on a misdiagnosis. It shouldn't guide the crew anymore, but destroying it would erase the incidents that seemed to confirm it. It stays in the archives. It's no longer available among the procedures authorized for flight.

I use the word "revocation" for this withdrawal of power. The belief stays connected to what supported it and to what challenged it, but its new status strips away its ability to govern a future situation.

This distinction sounds purely semantic until the agent can actually act. Writing "don't use invalid information" in a prompt amounts to entrusting the rule to the model. But the model is precisely the element the rule is meant to constrain. A guarantee that depends on its good faith is not a technical separation.

Cortex already does part of this work: a revoked learning is filtered out at recall time, then rechecked at action time. The refusal is named, timestamped, and persisted. But I have to be honest about the current limit: as long as the prohibition is declared to the model rather than enforced by code, it remains an instruction, not a guarantee. That's exactly what I'm working to harden.

07

Memory is a briefing, not an archive

Before a flight, captains don't receive the airline's entire archive. They receive a briefing prepared for that specific flight: destination, weather, aircraft status, known constraints.

Recall works the same way. The system selects admissible learnings and attaches the Episodes needed to understand them. It removes duplicates, preserves sources, and surfaces contradictions. If ten elements are relevant but capacity only allows five, the selection should say so rather than silently overflowing the model's context window.

A memory is not the entire archive. It's the part authorized to sit on the desk for this mission.

08

An open question

This work on memory had to come before something else.

Before Cortex, I tried to build an agent with guardrails inspired by fear, shame, guilt, and responsibility. Fear became risk anticipation, shame became a global rejection of incompatible results, guilt became a repair debt, and responsibility became a lock on action release.

That experiment stopped. For a debt to survive a session, for a past incident to change a future decision, or for a repair to remain owed, I first needed a memory capable of carrying that continuity.

I had built the guardrails before constructing the material that was supposed to feed them.

The work on memory now provides that material. It doesn't answer the next question yet, but it finally allows me to ask it properly: can past experiences create and sustain an artificial responsibility?

I build Cortex.deck, a system that gives AI agents continuity and experience. This piece is adapted from a longer essay, sourced and revised under an explicit human-AI collaboration protocol.