Memory Engineering and the Company Brain

Oluwasegun Araba, July 2026. Position paper. The architecture described here is dogfooded daily in a live operator workspace and generalised from production systems the author has shipped. The discipline claim is grounded in the cited literature. Where a claim is a position rather than a proven result, it is labelled as one.

ABSTRACT

Companies are wiring AI into everything and getting confident, wrong answers back. The reflex is to blame the model, or the vector database, or the framework, and to reach for a better one of each. This paper argues the failure is upstream of all of them. The knowledge the AI reads has no shape. Capture is accidental, storage is untyped, nothing is curated, and retrieval has no semantics, so a capable model returns fluent nonsense from disorganised source material. That is a memory engineering problem, not a model problem. The paper makes three grounded claims. The right response is not to fine-tune your knowledge into the model, because retrieval beats fine-tuning for injecting knowledge and an external memory can be inspected, edited, and owned. That memory should be a layered, managed system rather than one flat store. And none of this is new, organisational memory has been a defined discipline since 1991. From these we name the discipline, Memory Engineering, and give it a five-layer architecture. The honest concession is the frontier. Curating a living organisational memory at scale is unsolved, and there is no accepted metric for whether a memory layer is working at all.

1. The problem

Give a capable model a question about your business and it will answer with total confidence. Whether the answer is correct depends on something the model does not control, the state of the knowledge it retrieved to answer. In most companies that knowledge is a mess, and the model faithfully reports the mess back with the fluency of something that knows what it is talking about.

The reflex is to treat this as a model problem. A better LLM will fix it. A different framework will fix it. A second vector database will fix it. Teams re-platform their retrieval stack and get the same confidently wrong answers on top of a newer substrate, because they moved the plumbing and left the water dirty.

Look at where the knowledge actually lives. Wikis that decayed the week after launch. Slack threads with no structure. Decisions that exist in three people's heads and nowhere else. Post-mortems nobody wrote. Contracts, commitments, and the reasons behind past choices, all uncaptured or captured once and never maintained. Retrieval over that returns whatever is textually nearest, with no notion of which source is authoritative, how old it is, or whether two retrieved passages contradict each other.

This is not a modelling failure. It is an engineering failure in the layer beneath the model, the layer that decides what the organisation knows, in what shape, callable by whom, decaying how. That layer barely exists as a named responsibility in most companies. This paper is about that layer, what it requires, why the obvious shortcut is wrong, and the one part of it nobody has solved.

2. The wrong fix

When a team accepts that the model needs to know more about the business, the instinct is to put the knowledge into the model. Fine-tune it on the company's documents so it simply knows. It is an intuitive move and it is the wrong primitive, and here the evidence is not ours.

The distinction goes back to the founding retrieval work. Lewis and colleagues drew the line between two kinds of memory a model can use, parametric memory, the facts baked into the weights, and non-parametric memory, an external store the model retrieves from at answer time. Their motivation is the problem above stated precisely. A model stores knowledge in its parameters but cannot access or manipulate it cleanly, so on knowledge-heavy tasks it underperforms, and giving it a retrievable external memory closes the gap.

The head-to-head came later. Ovadia and colleagues compared the two approaches directly, fine-tuning a model on new knowledge versus retrieving it, on the same facts and the same tests. Retrieval won, consistently, and it won by the widest margin exactly where a business is most itself, the rare and specific facts. Models struggled to absorb new facts through fine-tuning at all.

State the transfer honestly. These are studies of knowledge injection into language models, not studies of a full organisational memory layer, and the leap from one to the other is an argument this paper makes rather than a result it cites. But the mechanism carries. Weights are a lossy, opaque, expensive place to put facts. You cannot look inside them to see what the model believes, you cannot edit a single fact cleanly, you cannot show a customer the source, and you cannot own the knowledge as an asset separate from the model you happened to train. An external memory is the opposite on every count. It is inspectable, editable, auditable, and yours. That is why the memory layer belongs outside the model, and why fine-tuning your company into a model is the confident wrong turn.

3. Memory is a managed system, not a store

Accepting that the memory lives outside the model raises the next question. What shape does it take. The weak answer is a single store, one big index you dump documents into and search. That is where most RAG deployments stop, and it is why they plateau.

The stronger answer treats memory as a managed system with structure and movement, and the clearest statement of it comes from MemGPT. Packer and colleagues borrowed the operating-system idea. A model has a small, fast main context, analogous to RAM, holding what it needs right now, and a large external context, analogous to disk, holding everything else, with the system paging information between the tiers and editing its own memory through defined operations. The point that generalises is not the specific two-tier design. It is that useful memory is tiered, with explicit rules for what lives where and how things move between levels, rather than one undifferentiated pool.

The honest limit. MemGPT designs its tiers for a single agent managing its own context window, not for an organisation's knowledge. The generalisation from one agent's memory hierarchy to a company's memory architecture is ours, offered as a design principle rather than a proven equivalence. But the principle is the load-bearing one. The moment memory is more than a bucket, you have to decide what gets captured, how it is typed and stored, how it is kept true over time, how it is retrieved with meaning, and how every surface reaches it. Those decisions are layers, and a memory system is the set of them working together.

4. An old discipline, newly urgent

There is a temptation to treat all of this as brand new, a category invented in the last two years alongside the models. It is not, and saying so is what separates a discipline from a pitch.

Company memory was given a rigorous definition thirty-five years ago. Walsh and Ungson, writing in the Academy of Management Review in 1991, defined organisational memory as a real construct, addressed the obvious objection that organisations do not literally remember, and laid out its structure and its three core processes, acquisition, retention, and retrieval. Those three processes are, almost exactly, what a memory layer does for a company. The discipline of managing what an organisation knows is older than every tool now used to do it, and any serious treatment of the subject inherits that lineage rather than pretending to have coined it.

The technical half has a pedigree too. Breit, van Harmelen and colleagues, surveying nearly five hundred papers in ACM Computing Surveys, mapped the field that combines machine learning with structured knowledge such as knowledge graphs and ontologies. They frame it through the neuro-symbolic split, learning as the intuitive system, structured knowledge and reasoning as the rational one, and real capability needing both. Combining a learning model with a structured, curated memory is not a workaround someone improvised. It is the direction a large, surveyed research field has been moving for a decade.

Use these two honestly. Walsh and Ungson predates modern AI entirely, so it grounds the legitimacy and the vocabulary, not any claim about how a language model behaves. The survey grounds the field, not any specific product. What the two together establish is the thing the reflex denies. This is a subject with a name, a history, and a research base, and it deserves to be engineered on purpose. We call that discipline Memory Engineering, the deliberate design of what an organisation knows, in what shape, callable by whom, decaying how.

5. The architecture

Memory Engineering has a reference architecture. Five layers, each answering a specific failure mode that produces a confidently wrong answer. The architecture below is our design, generalised from a memory discipline run daily in a live operator workspace and from production retrieval systems the author has shipped. It is a position, not a cited result, and the layers are the claim.

- **L1, Capture.** Knowledge that is never written down cannot be retrieved. The failure mode is no capture, the decision that lived in a meeting and evaporated. Capture has to be triggered, not voluntary. Decision forms, post-mortem templates, meeting digesters, a pull-request-to-decision-record pipeline, customer-call summarisers, the moments knowledge is created wired to capture it as a side effect of the work.
- **L2, Storage.** Captured knowledge with no type is a pile. The failure mode is wrong type, a policy stored as a chat message, a commitment stored as a loose note, indistinguishable at retrieval. Storage is a typed schema, every memory carrying its type, source, creation and verification dates, and links, versioned so you can ask what this said six months ago, and backed by something durable rather than a tool someone can delete.
- **L3, Curation.** Knowledge decays. The failure mode is stale, the confidently retrieved answer that was true last year. Curation is the discipline of keeping memory true, stewards owning domains, a promotion path from experimental to confirmed to canonical, and hygiene processes that surface what is stale, orphaned, or in conflict. This is the hardest layer, and section 6 returns to it.
- **L4, Retrieval.** Retrieval without semantics returns the textually nearest passage, not the true one. The failure mode is no retrieval semantics, and its uglier sibling, conflict swallowed, two sources disagreeing and the system silently picking one. Retrieval has to be typed and semantic, has to return provenance, source, age, and confidence, alongside the answer, and has to surface disagreeing memories together rather than hiding the conflict.
- **L5, Integration.** A memory only one tool can reach is a silo. The failure mode is a brilliant brain nothing is plugged into. Integration means every AI surface, the copilot, the Slack bot, the IDE, the onboarding portal, reads from the same memory, so the organisation has one brain rather than a different partial one behind each tool.

The layers are ordered because they depend on each other. Retrieval semantics are worthless over untyped storage. Curation has nothing to curate without capture. Building L4, the RAG layer, without L1 through L3 beneath it is the single most common mistake, and it is why so much deployed RAG returns fluent nonsense. The model was never the problem. The four layers under the retrieval layer were missing.

6. What it is not, and the open problem

Memory Engineering is easy to mistake for things it is not.

It is not knowledge management. That category is a graveyard of wikis nobody updates. Knowledge management asked people to maintain documentation as a virtue. Memory Engineering treats capture and curation as engineered, triggered processes, and builds memory to be read by machines as much as by people.

It is not RAG. RAG is one layer, L4, and building it alone is the failure this paper diagnoses.

It is not a wiki redesign or a vendor resale. The problem is not that your Notion is ugly. It is that your organisation has a capture, typing, and curation problem no tool fixes by default, and the layer sits above whatever database and model you already run rather than replacing them.

Now the honest frontier, in the spirit of naming what is not solved rather than letting an architecture diagram imply completeness. Capture is largely a matter of wiring. Storage and retrieval are engineering with known techniques. Curation is not solved. Deciding what is canonical versus stale versus in conflict,

across every domain of a living organisation, at the speed knowledge changes, without a human steward reading everything, is an open problem. The layered architecture bounds it. Stewards, promotion paths, and hygiene crons make it tractable. None of them automate the judgment at the centre of it.

Worse, there is no accepted metric for whether a memory layer is working. We can measure retrieval relevance and answer accuracy at the surface, but organisational memory quality, whether the company's knowledge is captured, current, and callable, has no agreed measure. A discipline without a metric for its own success is a young discipline, and honesty about that is part of taking it seriously.

That is where the work goes next. Automated curation of a living memory, and a real measure of memory health. Whoever solves those two moves the field.

7. Close

The industry is spending its attention on making models more capable. Capability is not the binding constraint for most companies trying to use AI. The binding constraint is that the model is reasoning over knowledge with no shape, and no better model fixes disorganised memory. It only reports it back more fluently.

Memory Engineering is the third leg of a discipline whose first two legs already have names. Context Engineering asks what should be in the model's context window for this task. Harness Engineering asks what scaffolding wraps the model, the tools, retries, evaluation, and agents. Memory Engineering asks what the organisation knows, in what shape, callable by whom, decaying how. Companies have started hiring for the first two. The third is still unnamed in most org charts, and it is the one that decides whether any of the AI built on top of it is right or confidently wrong.

The way in is small and concrete. Take the last five questions your AI got wrong, and map each to the layer that failed, no capture, wrong type, stale, no retrieval semantics, or a conflict swallowed. Every wrong answer has a memory failure mode behind it. That mapping is the whole diagnosis, and it is where the engineering starts.

References

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." Advances in Neural Information Processing Systems 33, 2020, pp. 9459-9474.
- Ovadia, O., Brief, M., Mishaeli, M., and Elisha, O. "Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs." arXiv:2312.05934, 2023.
- Packer, C., Fang, V., Patil, S. G., Lin, K., Wooders, S., and Gonzalez, J. E. "MemGPT: Towards LLMs as Operating Systems." arXiv:2310.08560, 2023.
- Walsh, J. P., and Ungson, G. R. "Organizational Memory." Academy of Management Review, Vol. 16, No. 1, 1991, pp. 57-91.
- Breit, A., Waltersdorfer, L., Ekaputra, F. J., Sabou, M., Ekelhart, A., Iana, A., Paulheim, H., Portisch, J., Revenko, A., ten Teije, A., and van Harmelen, F. "Combining Machine Learning and Semantic Web: A Systematic Mapping Study." ACM Computing Surveys, Vol. 55, No. 14s, Article 313, 2023.

The architecture is generalised from three production receipts. The Model Workspace Protocol, a memory discipline run daily on a solo operator's workspace since 2025, from which the five layers are generalised. invobi, a production multi-rail payments SaaS. And a live retrieval system serving a paying UK service business. That the specific five layers are optimal is a claim this paper does not make.