workerbee

WORKING PAPER 2.0

YOUR AI AGENTS HAVE A CONTEXT PROBLEM

Why enterprise AI needs a persistent understanding of the company beneath the agents.

Context engineering for the agentic enterprise

The problem is moving underneath the model

AI models are improving rapidly. Agents can now search, reason across multiple steps, call tools, query enterprise systems, and increasingly take action. For the first generation of enterprise AI, model capability was the obvious constraint: could the model understand the request, find the right information, and produce a useful answer?

That constraint is changing.

As models improve, a harder problem is becoming more visible: does the agent understand the organization it is acting inside? A generic model may understand what margin, customer health, qualified pipeline, or high performance mean in general. It does not automatically know what those concepts mean inside a specific company, which definition applies, how that definition has changed, or what evidence the organization trusts.

That distinction matters more as agents move from answering questions to making recommendations, initiating workflows, and influencing business decisions. Those activities require more than access to enterprise information. They require enough company-specific context to interpret that information correctly.

Google Cloud now describes business context and semantic meaning as increasingly important constraints on enterprise AI scale, alongside model capability itself. Its Agentic Data Cloud strategy reflects the same shift: from simply connecting agents to data toward giving them the business meaning required to reason over it reliably. [1]

The next enterprise AI problem is therefore not simply making agents more intelligent. It is giving them a reliable understanding of the business they are being asked to operate.

Does the agent understand the organization it is acting inside?

> **The models are getting better. The enterprise problem is moving underneath them.**

Enterprise AI is shifting from a model problem to a context problem.

ACCESS IS NOT UNDERSTANDING

Connecting an agent to more information does not give it a coherent model of the business.

Enterprise information already exists in enormous quantities. Structured data lives in databases and warehouses. Documents sit in content systems. Customer history is stored in CRM. Employee information lives in HCM. Policies, contracts, project history, conversations, and managerial judgment are distributed across many other systems and people.

Modern AI infrastructure is making more of that information accessible to agents. That is an important advance. But connectivity solves an access problem. It does not automatically solve an understanding problem.

A record may tell an agent that something exists without explaining why it matters. A policy may be retrievable without establishing whether it is current. Two systems may use the same term differently. Retrieved information may be factual, inferred, outdated, contradictory, or inappropriate for the decision being made.

This is why semantic context is emerging as a distinct architectural concern. Google, for example, positions its Knowledge Catalog as a context layer that aggregates enterprise knowledge, enriches it with business meaning, and makes that meaning available to agents through governed infrastructure. [2]

The distinction is simple. Access to a customer table does not tell an agent what makes a customer strategic. Access to an HR system does not tell it what success means in a particular role. Access to every document in the company does not tell it which information should influence the decision in front of it.

The enterprise usually has the information. What it often lacks is a shared representation of what that information means.

Access gives an agent information.
Context gives the information meaning.

THE RECONSTRUCTION PROBLEM

Every new agent should not have to understand the company again.

Much of enterprise AI still constructs understanding at the moment a question is asked. A request arrives. The application searches for relevant information, retrieves documents or database results, assembles context, and asks a model to interpret what it has found.

That approach is powerful. It allows an agent to work with information it was never trained on.

But it also means the company's understanding is often reconstructed one interaction at a time.

A second application may retrieve different evidence. Another agent may assemble the context differently. Prompts change. Models change. Sources are weighted differently. Useful context uncovered during one interaction may never become available to the next. The enterprise can therefore deploy increasingly capable agents while each independently recreates parts of the same organizational understanding.

The cost is more than duplicated computation. Repeated reconstruction can create inconsistent interpretation, higher latency, repeated inference expense, fragmented governance, and lost institutional knowledge. Google has begun describing similar problems as enterprises move toward agent scale: without consistent business context, reasoning can fracture across applications while costs and governance complexity increase. [3]

This creates a structural problem. The enterprise owns the underlying data, but much of the interpretation exists temporarily inside individual applications and model interactions. Once the interaction ends, the understanding that connected those pieces of information may disappear with it.

At small scale, that may be manageable.
At enterprise agent scale, it is an architectural flaw.

The better question is not how every agent can retrieve enough information to reconstruct the company. It is what understanding the company should preserve so every agent does not have to start over.

TODAY

Same company. Reconstructed three times.

The enterprise owns the data. The applications own the interpretation. The understanding repeatedly disappears.

workerbee

Prompt engineering became context engineering

For the first generation of applied AI, much of the engineering attention went into the prompt. What instruction should the model receive? How should the question be phrased? What examples or constraints would produce a better answer?

As agents become more capable, the problem is broader.

An agent may need instructions, tools, enterprise data, prior interactions, business definitions, memory, permissions, current task state, and the results of previous actions. The quality of the answer therefore depends not only on the prompt, but on the complete set of information available to the model when it acts.

Anthropic describes this shift as the move from prompt engineering to context engineering: curating and maintaining the information available to the model so it has the right context for the task. Anthropic also makes an important point that will matter throughout this paper: context is a finite resource. The objective is not to fill the context window. It is to make the information inside it as relevant and useful as possible. [4]

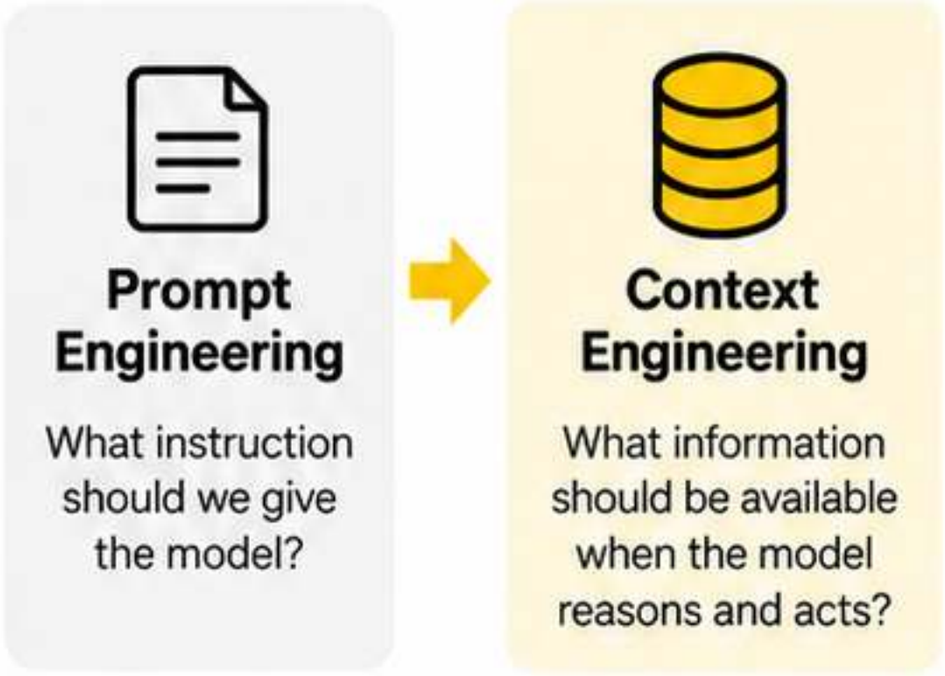
That changes the engineering question.

> **The first question was:**
> What should we ask the model?

> **The more important question for enterprise agents is becoming:**
> What should the model know when it acts?

> **And once that question moves from a single application to hundreds or thousands of agents, another follows:**
> Where does that company-specific understanding come from?

Prompt vs. Context



The context an agent needs to act

- Instructions
- Tools
- Enterprise data
- Business definitions
- Prior interactions
- Memory
- Permissions
- Current task state
- Results of previous actions

Better answers. Better outcomes.

The prompt tells the model what to do.
Context determines what it has to work with.

A bigger context window is not enterprise memory.

The intuitive response to missing context is to provide more of it: more documents, more database results, more conversation history, more prior interactions, more tokens.

Modern models can process increasingly large context windows. That is useful. But capacity is not the same as understanding, and a larger window does not turn transient model input into durable enterprise memory.

The limits are both technical and practical. Research has shown that models do not always use information in long contexts uniformly. Lost in the Middle found substantial variation in model performance depending on where relevant information appeared within a long input. [5]

Models have improved significantly since that research, but the architectural lesson remains. Anthropic still describes context as a finite attention budget and recommends finding the smallest set of high-signal information that maximizes the likelihood of the desired result. [4]

There is also an enterprise problem that context-window size cannot solve. Information supplied to a model for one task does not automatically become reusable organizational knowledge. The next agent may need to retrieve it again, reinterpret it again, and decide again what matters.

So the objective cannot simply be to give every agent more context.

It is to give each agent the right understanding when it needs it—and preserve what should not have to be relearned.

MORE CONTEXT ≠ ENTERPRISE MEMORY

Three concepts to keep clear.

Capacity =
How much can the model receive?

Relevance =
What should the model receive?

Memory =
What should the company preserve?

The answer is not to giwe every agent everything. It is to give each agent the right understanding when it needs it.

The challenge is architectural, not just technical.

The industry has proven that models can use large amounts of context at runtime. The remaining challenge is turning information into a persistent, shared, and correct understanding that multiple agents can rely on.

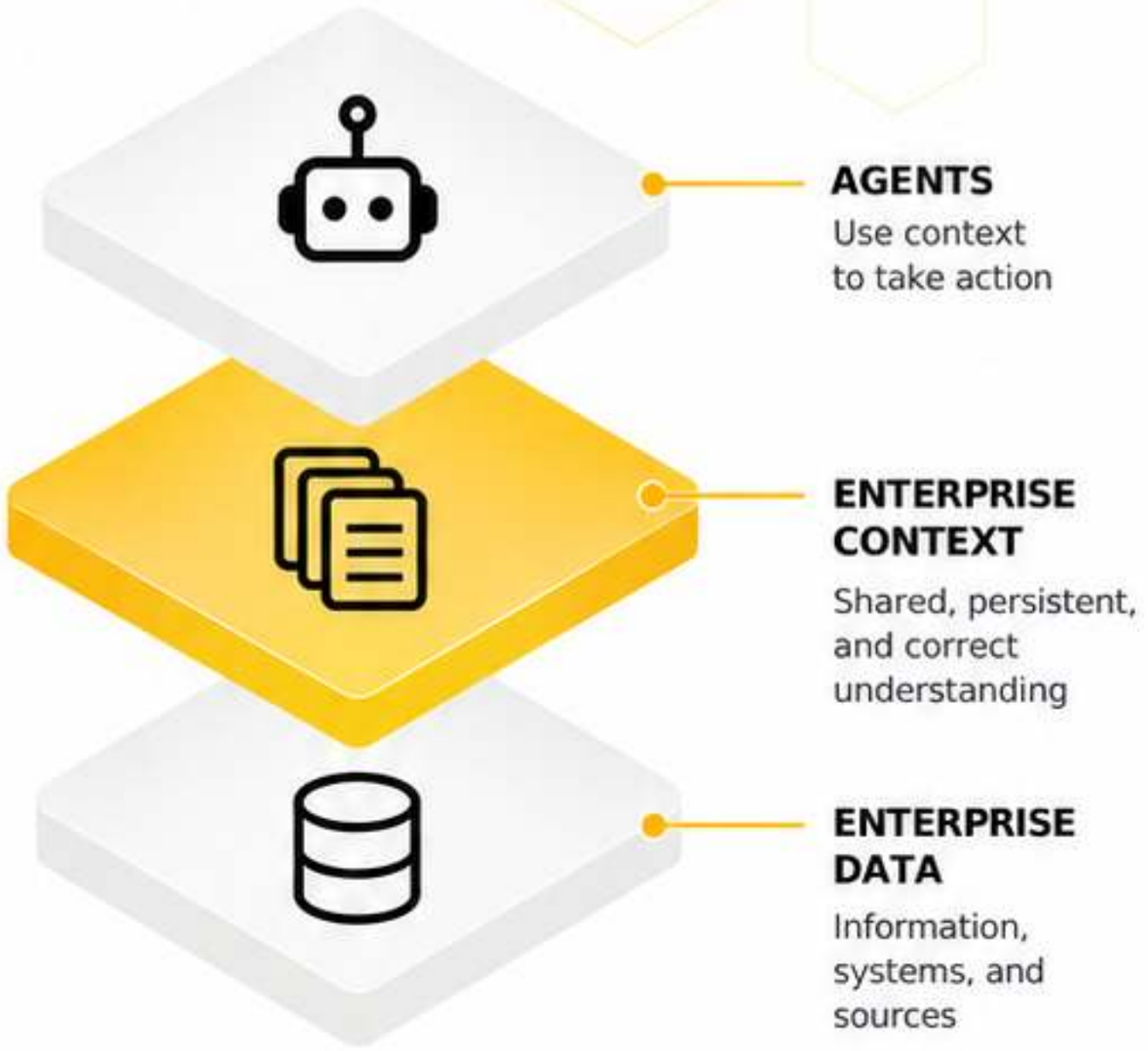
This is not simply a matter of storing more data. It requires decisions about representation, governance, quality, and change management.

Enterprise context must capture the meaning of information—entities, relationships, rules, definitions, and the constraints that give it validity. It must also evolve as the business changes, and remain consistent across agents and applications.

These challenges are familiar. They resemble the problems enterprises have faced for decades with data, metadata, and knowledge management. What is new is the immediacy. Agents need this foundation now, and the cost of not having it is higher because they act on information.

The question is no longer whether to provide more context. It is how to design, build, and operate an enterprise layer that delivers the right context to the right agents at the right time—and keeps it accurate, useful, and aligned with the business.

It is an architectural problem before it is a technical one.



IT'S THE FOUNDATION FOR ACTION

From information to understanding to impact.

CONTEXT TURNS INFORMATION INTO ACTION.

SHARED
Consistent understanding across agents and applications.

GOVERNED
Quality, validity, and control.

EVOLVING
Keeps up with the business.

Enterprise memory turns capability into continuity.

A larger context window helps a model work within a single interaction. Enterprise memory makes understanding persist across interactions, applications, and time.

Enterprise memory captures what matters, preserves it in a structured form, and keeps it available so it can be reused, combined, and built on.

It includes business entities, definitions, relationships, decisions, constraints, and evidence, along with their source, history, and provenance.

This is not just a technical improvement. It is an architectural foundation for agents that can operate reliably, explain their reasoning, and stay aligned with how the company works.

Without enterprise memory, every interaction is a fresh start. With it, each interaction contributes to a growing, organization-specific understanding that improves over time.

> **Enterprise memory is the foundation for agents that can operate reliably, explain their reasoning, and stay aligned with how the company works.**

CAPTURE
Identify and record what matters from interactions, documents, and systems.

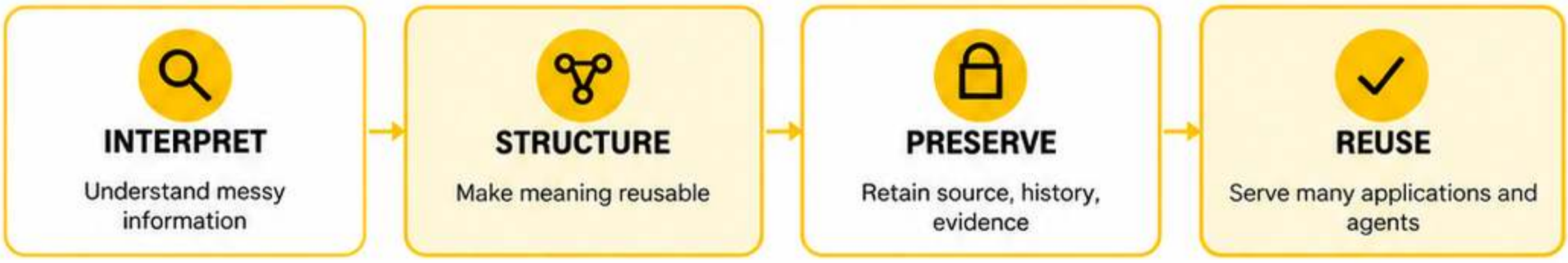
STRUCTURE
Represent information in a consistent, machine-readable form.

PRESERVE
Retain source, history, evidence, and provenance.

REUSE
Make it available for future interactions, applications, and agents.

Enterprise memory turns individual responses into lasting organizational knowledge.

Interpret once. Structure it. Preserve it. Reuse it.



Large language models are exceptionally good at interpreting information that was never designed for machines: documents, conversations, notes, policies, presentations, transcripts, and other forms of human knowledge.

That does not mean the model should have to reinterpret the same information every time the organization needs to use it.

A stronger architectural pattern separates interpretation from reuse.

Use models where interpretation is necessary: to identify relevant concepts, entities, relationships, evidence, constraints, and meaning inside fragmented information.

Then convert what matters into a representation that applications can use consistently rather than leaving all of that meaning trapped in raw text.

Preserve that understanding beyond the original interaction, along with its source, history, and supporting evidence.

And make it available to future applications, analyses, and agents without forcing each one to reconstruct the same meaning from the original material.

This principle has emerged directly from Workerbee's architecture work with Google Cloud and technical discussions with Google DeepMind. In Workerbee, Gemini helps interpret unstructured workforce information; BigQuery and BigQuery Graph support large-scale analysis across connected relationships; and the Work Graph preserves the company-specific understanding required for repeated decisions.

Those discussions have also pushed the harder questions beneath the architecture: what deserves to become structured knowledge, what should remain evidence, how contradictions should be preserved, and how provenance, time, confidence, and graph quality should be maintained.

The specific implementation is Workerbee's.

The underlying principle is broader:

In practice: Workerbee separates model interpretation from persistent structured evaluation—an architectural pattern developed and tested through its work with Google Cloud.

> **Models should help create enterprise understanding. They should not be the only place that understanding exists.**

Working context and persistent enterprise context are not the same thing.

WORKING CONTEXT

An agent needs temporary information to complete a task.

What did the user just ask? What tool returned a result? Which step of the workflow is underway? What intermediate conclusion needs to be checked?

That is working context.

It helps the agent reason through the task in front of it. Much of it may matter only for seconds or minutes.

PERSISTENT ENTERPRISE CONTEXT

Persistent enterprise context has a different lifecycle.

A customer definition. A contractual relationship. An organizational policy. A validated capability. A product dependency. A business rule. A prior decision and the evidence behind it.

Those may need to remain usable across applications, agents, and years of organizational activity.

WORKING CONTEXT

- Temporary
- Task-specific
- Session state
- Intermediate results
- May be discarded

PERSISTENT ENTERPRISE CONTEXT

- Reusable
- Company-specific
- Source-linked
- Governed
- Maintained over time

The distinction matters in both directions.

Durable company context should not disappear because an agent session ended. But temporary observations and model-generated conclusions should not silently become durable organizational truth.

Modern enterprise architectures are beginning to make this separation explicit. Microsoft, for example, describes its ontology layer as a governed business model where concepts, relationships, rules, and source bindings are defined once and reused across agents and workflows. [9]

Workerbee has encountered the same problem in practice. Continuously maintained company context requires controls around source, time, confidence, conflicting evidence, and what is permitted to enter persistent structure at all.

That leads to a simple rule:

Persistent enterprise context should outlive the agent session. Working context should not automatically become enterprise truth.

> **Working context helps an agent complete a task. Persistent context helps the company remember what it knows.**

In high-stakes domains, context must be governed.

The architecture problem becomes more important where decisions carry real consequences.

In some enterprise settings, a weak answer is inconvenient. In others, it is consequential.

When an AI system influences who gets hired, which employee gets promoted, what care pathway a patient follows, which customer receives credit, or what action is taken in a regulated process, the context supplied to that system cannot be left to chance.

It is not enough for the model to retrieve something relevant. The organization must also know whether the information was appropriate, current, authorized, consistently applied, and defensible for the decision being made.

This is the point where context engineering becomes governance.

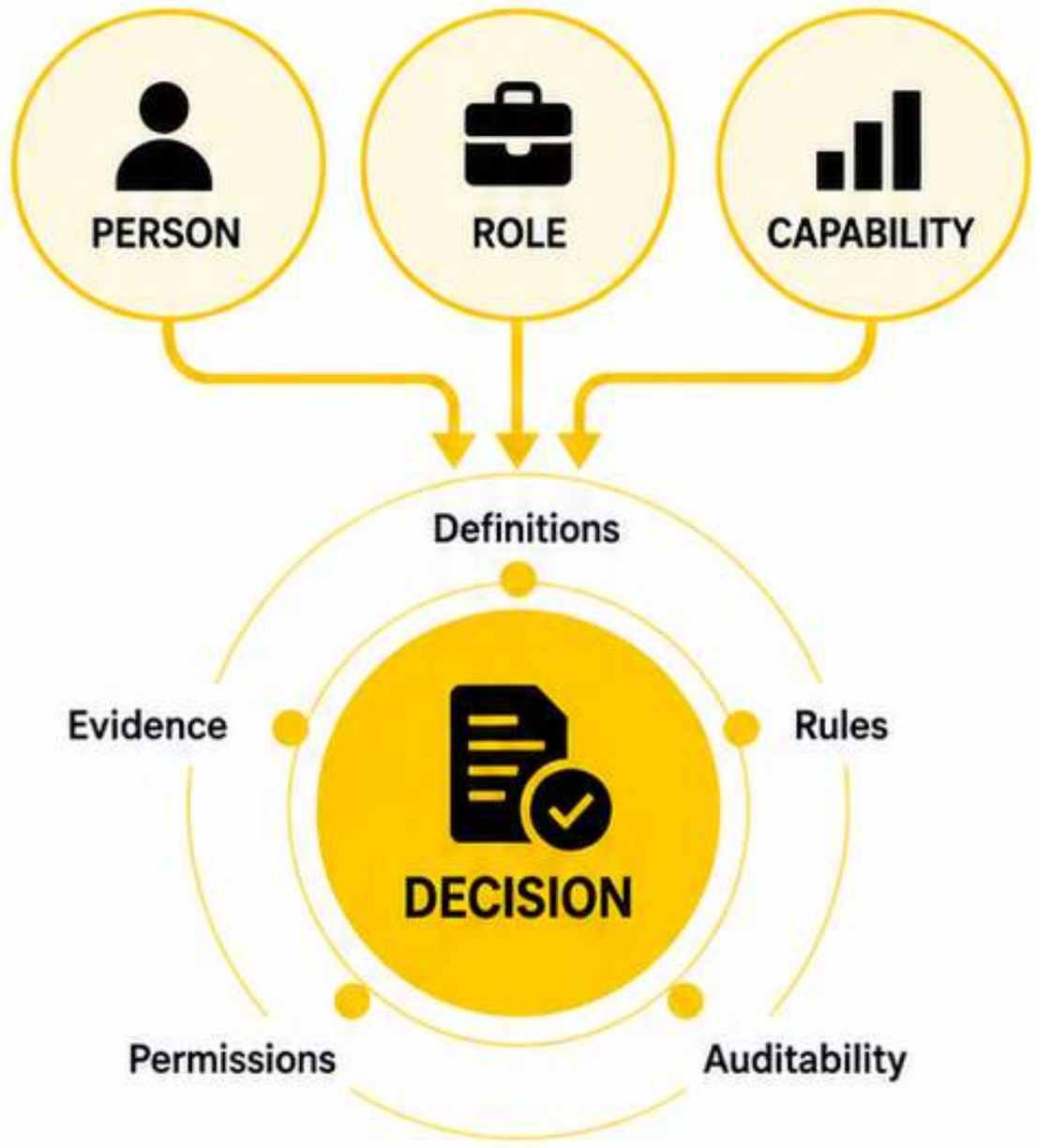
A company needs to decide what definitions are authoritative, what evidence can be used, what rules apply, what context should persist, who can modify it, and what downstream systems or agents are permitted to rely on it.

In lower-stakes workflows, inconsistency may create inefficiency. In higher-stakes workflows, inconsistency creates risk.

This is one reason governed semantic layers, knowledge graphs, ontologies, and context platforms are drawing more attention. They do not just make AI systems smarter. They create a controllable layer of business meaning that can be reviewed, maintained, and trusted.

Workerbee is built for this kind of environment in workforce decision-making. Hiring, internal mobility, development, and workforce allocation are all areas where context influences outcomes for real people. In those settings, the context itself becomes part of the control surface.

> **In high-stakes domains, context is not just helpful. It is part of the governance model.**



In consequential decisions, the question is not only whether the system can act. It is whether the organization can trust the context it acted on.

Persistent context can make an agent better. Persistent bad context can make it reliably wrong.

Once enterprise context becomes persistent and reusable, a harder question appears:

What is allowed to become part of that understanding?

Enterprise information is not uniformly trustworthy.

It can be outdated, incomplete, contradictory, inferred, incorrectly attributed, permissioned, or simply wrong. A model can also generate a plausible interpretation that should never become organizational knowledge without review.

Persistence therefore cannot be the goal by itself.

A governed context layer needs to preserve more than a conclusion. It needs enough information to answer basic questions about the knowledge beneath the decision:

SOURCE
Where did it come from?

STATUS
Fact, inference, or judgment?

TIME
When was it true?

ACCESS
Who may use it?

EVIDENCE
What supports it?

CHANGE
What changed, and who approved it?

This is consistent with broader AI risk-management guidance. NIST's AI Risk Management Framework emphasizes documentation, data provenance, intended context of use, human oversight, monitoring, and controls around the information and systems used by AI. [11]

Enterprise platforms are beginning to incorporate similar principles into their context architectures. Microsoft's ontology model, for example, combines governed definitions, source mappings, provenance, and access controls so agents reason from a shared business model rather than independently interpreting raw data. [9]

Workforce decisions provide a demanding test of the same principle. In Workerbee's architecture work, provenance, temporal reasoning, contradictory sources, confidence, protected information, and controlled updates are treated as part of knowledge formation itself—not as governance added after the answer.

That leads to the larger principle:

The more consequential the decision, the more governed the context beneath it must be.

Persistent enterprise context is not simply memory.

It becomes decision infrastructure.

> **As agents move from answering questions to influencing consequential decisions, context governance becomes as important as model governance.**

Better context is also an economic advantage.

The architecture is not only better for reasoning. It is better for cost.

If the enterprise repeatedly reconstructs the same understanding inside prompts, retrieval steps, and agent workflows, it pays for the same reasoning again and again.

That cost appears in several places: repeated retrieval, repeated prompt assembly, repeated model inference, repeated latency, and repeated engineering effort to rebuild similar context across applications.

This is one reason context engineering matters economically. The more context a company can structure once, preserve, and reuse appropriately, the less often it has to recompute the same understanding from scratch.

This does not mean all reasoning disappears. It means more of the expensive work moves upstream: deciding what meaning should persist, what relationships matter, what evidence should be carried forward, and what context can be reused safely.

That architectural shift has direct financial implications. Research and vendor guidance increasingly emphasize cost control through context quality, retrieval discipline, caching, and reducing unnecessary token consumption. [11] [12] [13]

Workerbee sees the same pattern in workforce intelligence. If every hiring, mobility, or workforce-planning question has to rediscover the same role context, capability definitions, and evidence structures, AI costs rise while consistency falls. If that context is preserved and governed, the same foundation can support many decisions.

A MORE EFFICIENT APPROACH

- **STRUCTURE ONCE**
- **PRESERVE**
- **REUSE**

versus:

A MORE EXPENSIVE APPROACH

- **RETRIEVE AGAIN**
- **REASSEMBLE AGAIN**
- **REASON AGAIN**
- **PAY AGAIN**

> **The most expensive token is often the one spent recomputing what the company already knows.**

The economic case for enterprise context is not only about better outputs. It is about lowering the cost of producing them repeattedly.

Governance has to reach beneath the model.

In many enterprise applications, imperfect context creates an inconvenient answer.

In others, it can influence a consequential decision.

A hiring recommendation can determine who receives an opportunity. A promotion recommendation can change a career. A redeployment decision can determine whether someone remains with the organization. Capability assessments can affect who is developed, retained, or overlooked.

In those settings, the context beneath an AI recommendation cannot simply be whatever an agent happens to retrieve.

It has to be deliberately governed.

That means controlling which information is permitted to influence the decision, which attributes are excluded, which company-specific standard applies, whether information represents fact or inference, how current it is, what evidence supports it, who reviewed important changes, and how the resulting recommendation can later be reconstructed.

This is where the distinction between model governance and context governance becomes important.

Organizations increasingly understand that models need controls: testing, monitoring, permissions, human oversight, and defined use cases.

But a well-governed model operating on poorly governed context can still produce a bad decision.

Workerbee was built around this problem in workforce decisions. The system separates the company-specific Success Profile, underlying evidence, structured evaluation, recommendation, human decision, override, and eventual outcome rather than collapsing them into a single generated answer. Its learning model is also explicitly controlled: new knowledge and relationships are reviewed before becoming part of the reusable system.

The point is not that workforce decisions are unique.

It is that they make the consequences of weak context unusually visible.

That produces three progressively harder questions:

That last question becomes more important as the consequences rise.

> **A well-governed model operating on poorly governed context can still produce a bad decision.**

CACHING

How do I avoid processing the same context again?

PERSISTENT CONTEXT

How do I stop reconstructing the company every time?

GOVERNED CONTEXT

What context is permitted to influence this decision?

The system gets smarter because the company gets clearer.

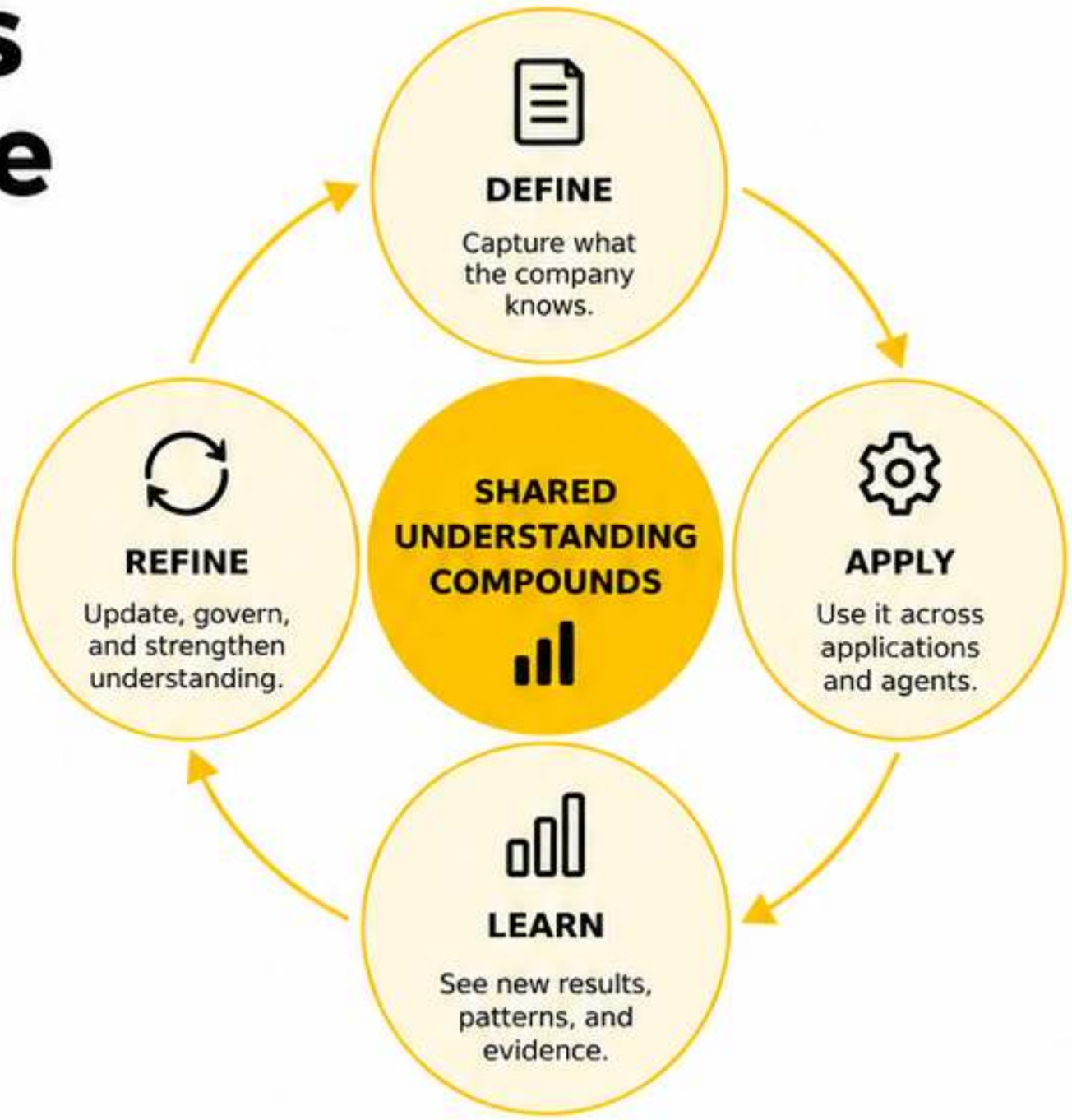
The long-term value of enterprise context is not just that it improves a single answer. It is that the organization can refine and reuse what it learns.

A company may change how it defines strategic customers. It may learn which signals best predict success in a role. It may discover which capabilities transfer effectively into adjacent work. It may determine which decisions require tighter governance and which can be handled more autonomously.

If that understanding remains trapped inside prompts, isolated workflows, or one-off model interactions, the organization has to rediscover it repeatedly.

If it is preserved, governed, and exposed as reusable enterprise context, the company can improve the quality of decisions over time while reducing repeated interpretation work.

This is one reason the context layer matters strategically. It does not just support better agent behavior. It helps the company accumulate institutional understanding that compounds.



Institutional understanding should compound.

Why Workerbee is writing about enterprise context

Workerbee did not start with a product idea or a market strategy.

We started with a simple conviction: enterprise work will only move forward by solving a more fundamental problem. Most of the AI conversation is about models and agents, but the real constraint is context. Without a persistent, accurate and usable understanding of the company, agents will stall, repeat mistakes, and ultimately fail to deliver value. Enterprise context is the operating system.

Where should companies spend their understanding tax when many models, applications, and agents need to use it?

We have also learned that this is not just a technical challenge. Context sits at the intersection of data, people, process, and trust. It touches every part of the business and requires a new approach to governance, architecture, and change management. That is why we are writing about it now.

We believe context is the missing piece for making AI useful at work. In this paper, we have shared what we have learned so far and a framework for how enterprises can think about it. We do not claim to have all the answers. Instead, we see this as an invitation to the community of builders, operators, and business leaders to engage, challenge, and improve these ideas.

Our goal is to help move the conversation forward, from models and hype, to the real foundations of enterprise AI: context, understanding, and people who can put it to work in meaningful ways.

Because when enterprise context works, agents work. And when agents work, the real opportunity of AI inside the enterprise becomes possible.

That is why we are writing.

> **"As agents move from answering questions to executing meaningful business work, persistent enterprise context becomes their most important operational asset."**

ABOUT THE AUTHORS

Parker Beek
Founder & CEO,
Workerbee

Matthew Paananen
CTO

Working Paper 2.0
September 2025

REFERENCES

[1] OpenAI. (2024).
GPT-4 Technical Report.
OpenAI. Details the model architecture and capabilities of GPT-4, including emergent behaviors and limitations.

[2] Anthropic. (2024).
Claude 3 Model Card.
Anthropic. Provides model specifications, benchmark results, and guardrail strategies for Claude 3.

[3] Google. (2023).
Gemini Technical Report.
Google DeepMind. Describes the Gemini model family, with a focus on multi-modal capabilities.

[4] Microsoft. (2024).
The New Bing and Enterprise AI.
Microsoft. Discusses enterprise integration, security, and governance for LLMs in the workplace.

[5] Liu, et al. (2023).
"Lost in the Middle: How Language Models Use Long Contexts."
arXiv. Analyzes how LLMs process and attend to information in long documents.

[6] Lewis, P. et al. (2020).
"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks."
NeurIPS. Introduces retrieval-augmented generation (RAG) as a method to ground model outputs in external knowledge.

[7] Model Context Protocol. (2024).
Model Context Protocol Specification.
Anthropic. An open standard for connecting AI systems to data, tools, and enterprise systems.

[8] LangChain. (2024).
LangChain. Documentation.
LangChain. Provides tools and frameworks for agent and workflow development.

[9] Pinecone. (2023).
Vector Databases for Enterprise AI.
Pinecone. Explains vector database architectures and best practices for RAG systems.

[10] Gartner. (2024).
'Hype Cycle for Generative AI.
Gartner. Analyzes the maturity and adoption trajectory of generative AI technologies in the enterprise.

[11] McKinsey. (2024).
"The Economic Potential of Generative AI.
McKinsey & Company. Estimates the business impact and economic value of generative AI across industries.

[12] Harvard Business Review. (2023).
"How Organizations Can Build Trust in AI.
HBR. Discusses governance, transparency, and human oversight as critical components of successful AI adoption.

WORKERBEE SOURCE NOTES

References to third-party publications, tools, and proprietary examples in this paper are drawn from Workerbee benchmark testing and its own analysis and interpretation.

Workerbee is not affiliated with any of the referenced entities. They are cited for informational use only and do not necessarily constitute endorsements.