

AI LEARNS / CONTEXT WINS.

Why the future of AI depends on something AI cannot create.

By Frédéric ROBERT, Advisory Director

4 downloadable
infographics

25 min
read

30 pages
expert insights

Content

01 A debt twenty years in the making p 5

POV page	p 6
Chapter 1 - The Fracture Was Already There	p 7
Chapter 2 - The Hidden Enterprise Debt Of Isolated AI	p 8
Chapter 3 - What Enterprise Context Is Not	p 9

02 Beyond The Context Layer Gold Rush p 10

POV page	p 11
Chapter 1 - Everyone Wants to Own the Name	p 12
Chapter 2 - Enterprise Context is Not Another Platform	p 13
Chapter 3 - Built to Outlast the Stack	p 14

03 The Context Exists. It Just Isn't Connected p 15

POV page	p 16
Chapter 1 - What We Mean by Enterprise Context	p 17
Chapter 2 - Built for Data, Not for Context	p 18
Chapter 3 - Ten Components, One Governed Asset	p 19
Chapter 4 - Human Expertise Creates Tomorrow's Context	p 20

04 Governance Is the Missing Layer p 21

POV page	p 22
Chapter 1 - The Definition Needs an Owner	p 23
Chapter 2 - Trusted Foundations Before Trusted Context	p 24
Chapter 3 - The Enterprise Context Broker	p 25

05 Where Enterprise Context Actually Starts p 26

POV page	p 27
Chapter 1 - Start Narrow, Govern Deep	p 28
Chapter 2 - Three Streams, Not One Road Map	p 29
Chapter 3 - Certify Once, Improve Forever	p 30

CONCLUSION

Executive Summary

Every enterprise investing in AI is about to discover the same thing.

The model was never the problem.

AI is not failing because organizations lack data. Most have more than they can use. It is failing because nobody agreed on what that data means: the ownership, the rules, the exceptions that turn a number into something trustworthy. A machine cannot ask. For years, a human closed that gap quietly, reconciling three customer IDs, deciding which "active customer" applied this quarter. AI doesn't pause to ask. It executes on whatever it retrieves first, and reports the result as fact.

The instinct is to buy the fix. A catalog. A knowledge graph. Another RAG assistant. Each one ships fast, proves value in its demo, and rebuilds definitions the last project already resolved: three assistants, three versions of "active contract," disagreeing in production. organizations won't scale AI by deploying more of them. They'll scale it by governing the context every assistant draws from, and by capturing the knowledge no assistant can generate on its own: built once, certified once, reused everywhere.

Enterprise Context isn't a platform to procure. It's a governed capability, built from master data, metadata and governance the organization already owns, connected for the first time instead of managed in silos. AI exploits existing knowledge.

Human expertise keeps creating new knowledge nobody has written down: the exception a manager approved and never revoked, the judgment call an analyst repeats every quarter without anyone asking why.

Enterprise Context Management connects both, before scale turns every unresolved definition into a decision no human reviewed.

AI is becoming a commodity. Enterprise Context will not.
That is where the advantage will actually sit.

Everybody talks about the context layer. Almost nobody means the same thing.

I have sat in this meeting a dozen times this year. A CDO opens a vendor deck, and the words "context layer" appear on the third slide, promising to fix everything from hallucination to governance debt. Nobody in the room agrees on what the term means. Nobody asks.

Everybody talks about the context layer. Almost nobody means the same thing.

A catalog vendor calls it metadata. An MDM vendor calls it a golden record. A RAG vendor calls it grounding. Each is technically correct. None answers the question the enterprise actually has: who decides which definition holds, which exception still applies, whether the answer can be trusted today.

That question is not new. It has sat unresolved for twenty years, absorbed quietly by people who reconciled the mess before anyone had to name it.

This paper is not another attempt to name the winning platform. It introduces Enterprise Context Management for what it is: a discipline, not a product, built from what most enterprises already own and have never connected.

I have watched too many strong AI investments fail for a reason nobody diagnosed correctly. **This paper makes the diagnosis explicit.**



Frédéric ROBERT





PART
01

A DEBT TWENTY YEARS IN THE MAKING

“AI didn't create the fracture. It just removed the human being who used to hide it.”

Twenty years of data management produced governed assets. It never produced a governed answer to the question that sits above them: which definition applies, right now, for this decision. Nobody noticed, because somebody was always answering it by hand.

Organizations don't have a data problem. They have an unresolved argument about what their data means, one a human always quietly closed.

AI didn't create that argument. It just made it impossible to keep ignoring.



1. The Fracture Was Already There

Why has context fragmentation always limited enterprise decision-making?

The same customer appears under three identities: one in CRM, one in billing, one in analytics. Every large enterprise has some version of this. For twenty years, a human sat between that fragmentation and the decision, quietly reconciling it before anyone noticed.

AI removes that buffer.

Most enterprises believe they solved this already. MDM, metadata, governance and knowledge management each covers a piece of it, built and owned separately, rarely talking to each other. That was tolerable when a person absorbed the gap. A case manager could ask a colleague which definition of "active client" applied this quarter. An analyst could reconcile the three customer IDs before writing the report. A supplier risk score could sit unused for a week until someone with context decided whether it still applied.

An AI agent does not ask. It executes on whichever version it retrieves first, and produces a decision that looks confident and is wrong.

Nobody put a number on this shift, because nobody had to. The organization was never arguing about the data. It was arguing about what the data represents, and it has been arguing for two decades without calling it that. AI did not create that disagreement. It removed the only mechanism that used to hide it.

The pattern repeats at every layer: two supplier codes for the same vendor, conflicting ownership fields that nobody has resolved because no single role was ever accountable for the resolution. None of this is new. What is new is a system fast enough, and confident enough, to act on the conflict without pausing to notice it exists.

Today, a human reconciles the fracture, quietly, one judgment call at a time. Tomorrow, AI automates the inconsistency, at a speed and a scale no human can review after the fact.

What has changed is the cost of leaving it unresolved. A misaligned dashboard produces one slow, forgivable mistake, caught by a human who senses something is off. A misaligned agent produces the same mistake thousands of times before anyone reviews the output, and by then it has already shaped a decision, a report, or a customer interaction.

KEY TAKEAWAY

The next maturity step is not more data management. It is connecting the disciplines the enterprise already owns into one governed layer of shared meaning, before AI turns every unresolved definition into an operational decision.



2. The Hidden Enterprise Debt of Isolated AI

Why does 'one RAG per use case' create duplication and governance risk?

One team builds an assistant for customer service. Another builds one for legal documents. A third builds a knowledge graph for product data. Each project ships fast, proves value in its demo, and gets funded for the next phase. This looks like three successful pilots.

It is actually three versions of the same unresolved definitions.

Each one also rebuilds the same context from scratch. The customer service assistant indexes the returns policy. So does the legal assistant, a slightly older version, pulled from a different folder. Two definitions of "active contract" now live in two separate systems, answering two separate questions, disagreeing with each other quietly, in production.

Each assistant carries its own retrieval logic, its own access rules, its own maintenance burden, and no one accountable for reconciling them with the others.

The debt is invisible at first because each assistant works, on its own, for its own use case. It becomes visible the moment two assistants answer the same question differently in front of the same executive.

That is not a model problem. It is context that was never governed once, only rebuilt repeatedly, at increasing cost and decreasing trust.

Governing it once doesn't mean forcing every assistant to look identical. Some local adaptation will always be necessary: a legal assistant needs context a customer service bot does not. But the definitions, the policies, the entities both depend on should exist once, governed once, and get consumed many times, not reinvented behind separate closed doors.

The organizations furthest ahead are not the ones with the most AI assistants. They are the ones where the fourth one took two weeks to build instead of two months, because the context it needed already existed, certified, reusable.



KEY TAKEAWAY

The goal is not one context stack per use case. It is one governed context, factorized once, reused at scale across every assistant that needs it.

3. What Enterprise Context Is Not

Why was the technology already in place never designed to answer this?

Every contact center handles this call daily: a customer asks for the loyalty discount. Someone, or something, needs to answer: are they eligible, today, right now?



A **RAG system** retrieves the loyalty policy document and quotes the eligibility threshold with confidence, unaware it was superseded three weeks ago or that a regional exception applies to this account. It answers what the text says, not whether the text is still true.



A **knowledge graph** shows the relationship instantly: this customer, this product, this loyalty tier, traversable in milliseconds. What it doesn't carry is the rule that governs the exception, or who can approve it. It knows the shape of the relationship, not the decision inside it.



A **semantic layer** defines "active customer" with total precision, everywhere it's queried. Precision is exactly its strength, and its limit: it has no mechanism for the regional exception, because an exception is a judgment call someone made once and never wrote down anywhere the semantic layer could reach.



The **catalog entry** does its job quietly: the field exists, an owner is named, a timestamp confirms the last update. Nobody asked whether that means the value can be trusted today.

Four technologies, four correct answers, and the customer is still waiting.

None of them is the wrong tool. None was built to resolve a decision. Each was built to answer a narrower question well, in isolation from the other three.

The eligibility answer only exists where retrieval, relationship, definition and ownership converge, the exception folded in, the freshness confirmed. No single technology sits at that convergence point. That is not a gap in any one of them. It is the space between all of them, and it is exactly what remains ungoverned.

KEY TAKEAWAY

Enterprise context is not retrieval, not a graph, not a definition, and not a catalog entry. It is fragmented meaning, fragmented ownership and fragmented memory, converging into one decision, and someone accountable for making sure they agree.





PART
02

BEYOND THE CONTEXT LAYER GOLD RUSH

“The Enterprise Context Layer isn't a platform to buy: it's a capability to govern.”

Every technology category wants to own the label, because whoever owns it owns the budget line. That competition produces RFPs promising full coverage and demos that all look complete, yet none of it resolves the real question. Metadata, MDM, knowledge graphs, RAG and semantic layers each govern one fragment of context correctly, but none decides which fragment to trust, when, or for whom.

The organization does not need a fourth vendor claiming completeness. It needs its own definition of enterprise context, built before any purchase, not supplied by whichever roadmap slide shouts loudest.



1. Everyone Wants to Own the Name

Why is nobody incentivized to agree on what "context" means?

Ask ten vendors what the Enterprise Context Layer is, and ten of them will describe their own product. The data catalog vendor calls it a catalog. The MDM vendor calls it a golden record. The RAG and vector-search vendor calls it grounding. Each answer is coherent on its own terms. None of them agree with each other. **This is not confusion.**

It is positioning. Whoever owns the label gets to define the budget line, and a market that cannot agree on what a term means is not failing to communicate. It is competing for a category, and category ownership is worth more than any single feature.

In practice, this leaves the CDO holding three RFP responses that each claim full coverage of "context," built on three incompatible architectures, none of which use the term the same way. Every vendor produces a slide where their platform sits at the center and everything else is labeled "integration." Buyers are told that once this one platform is in place, the context problem is solved. It rarely is: the RFP resolves a purchase, not a capability.

A March 2026 Gartner analyst note* found that vague AI claims now trigger scrutiny rather than confidence, and the same discounting is starting to apply to "context." A term that once signaled clarity now fits almost any roadmap slide, which means, on its own, it no longer signals anything.

Concretely, this means the RFP is the wrong starting point.

TECHNOLOGY CATEGORY	WHAT IT GETS RIGHT	WHAT IT DOESN'T SOLVE ALONE
 RAG & Vector Search	Retrieval, grounding	Whether the retrieved answer is still current
 Knowledge Graph	Relationships, semantics	Which rule applies, and who can authorise it
 Semantic Layer	Business metrics, KPI logic	Exceptions that were never written down
 Metadata & Catalog	Meaning, lineage, ownership	Whether the value can still be trusted today
 MDM	Identity, reference entities	Business rules and institutional memory
 Knowledge Management	Documents, expertise	Structured, governed, machine-readable access

The starting point is defining what enterprise context is, independently of any vendor's definition of it, so every technology can be evaluated against a capability the organization owns rather than a category the market is still fighting over.

KEY TAKEAWAY

The market's problem was never disagreeing on the name. It is that every category profits from the disagreement continuing, and the table above proves it: six technologies, six correct answers, none sufficient alone. Enterprise Context Management is the discipline that connects, governs and exposes all six as a shared enterprise asset, the definition no RFP built on a vendor's roadmap will hand the organization for free.

*Source: Sabine Zimmerhansl, Analyst Take: AI Washing Is Backfiring, Gartner, 23 March 2026, ID G00850823

2. Enterprise Context is Not Another Platform

If it's not a platform, what exactly is being governed?

Retire the RFP, and a different question appears immediately: if no platform is the answer, what exactly is being governed. Enterprise context is the governed set of meanings, rules, ownership and memory that lets a person, a report or an AI agent interpret data correctly and know whether it can still be trusted today. Metadata, master data and governance each covers part of that.

None of them, alone, covers all of it.

A supplier risk score sits in a dashboard reading eighty-seven, high risk. The number is accurate. It does not say which year's methodology produced it, whether the supplier disputed the result, or who is authorized to override it before the contract renews. The catalog entry, the master record and the governance policy each hold a fragment of that answer.

Enterprise context is what assembles the fragments into one, before the renewal decision gets made on a number nobody has re-checked.

That is why it behaves like a capability rather than a system.



A platform is procured, deployed, and marked done.



A capability is staffed, resourced and governed continuously, because the business keeps producing new exceptions, new definitions and new ownership changes faster than any static system can absorb them.

This has a direct consequence for who owns it. A platform gets an IT budget line and a go-live date. A capability needs an accountable owner, a standing operating model, and a mandate that survives the person who first championed it.

Most of what that capability needs is not missing. The definitions exist somewhere. The ownership records exist somewhere. The quality signals exist somewhere, in the catalog, the MDM system, the governance policy, and the people who already reconcile the gaps by hand.

What is missing is the governance that connects it into one asset instead of leaving it scattered across four.

KEY TAKEAWAY

Enterprise context doesn't replace the catalog, the MDM system or the semantic layer. It orchestrates them into one governed asset instead of four disconnected ones.

3. Built to Outlast the Stack

What happens when the platform underneath it changes?

Somewhere in the enterprise, a definition of "active contract" was rebuilt last quarter for the third time in four years. Not because the definition changed. Because the catalog moved, then the retrieval method, then the model. None of this is a warning about the future. It is what already happened to most of the technology compared two pages back.

And new models now ship faster than most governance teams can review them.

This creates a question most context initiatives never ask before they start: what happens to the context itself when the tool holding it disappears?

If the answer is "it gets rebuilt," the organization has not governed context. It has rented it, one platform cycle at a time, paying the reconstruction cost every time that cycle turns over.

Enterprise Context Management treats that question as the design constraint, not an afterthought.

A definition, a rule, an ownership record or a certified piece of memory needs to survive the retrieval method, the vendor and the model that happened to expose it this year. That means describing and governing context independently of any single technology, portable enough to move to whatever consumes it next, an agent framework, a different LLM, a platform nobody has built yet, without being reinterpreted from scratch.

As foundation models become more interchangeable across vendors, they stop being where competitive advantage lives. An organization that rebuilds its context every time it swaps a platform is competing on the same commodity everyone else has access to. What doesn't standardize is the enterprise's own governed context.



Vendor
acquired



Architecture
replaced



Model
outdated



Governed context
persists

KEY TAKEAWAY

Technology changes on its own schedule. Enterprise context should not have to change with it. A context layer worth governing is one built to move to the next platform, not rebuilt every time one arrives.

PART
03

THE CONTEXT EXISTS.
IT JUST ISN'T CONNECTED.



“The organization isn't missing context. It's sitting on top of it.”

Every enterprise assumes its context gap is a collection problem: more documentation, more definitions, another catalog. It rarely is. Master data, metadata, governance rules, project knowledge and the judgment calls in people's heads already contain most of what AI, analytics and decision-makers need.

What's missing is the discipline that connects, governs, qualifies and exposes it. Enterprise Context Management doesn't start with a new investment. It starts with the ones already made, treated for the first time as one asset instead of five disconnected ones.



1. What We Mean by Enterprise Context

Two words, how many definitions?

Five teams use the word "context" in the same meeting and none of them stops to check they mean the same thing. Nobody is wrong. Nobody is aligned either..



Enterprise context is the governed set of meanings, relationships, rules, ownership, memory and operational signals that lets a person, a report or an AI agent interpret data correctly. Not the data itself, but what the data represents, which rule applies to it, who is accountable for it, and whether it can still be trusted today.



Enterprise Context Management is the discipline that makes that context explicit, owned, certified and reusable. It is not a tool, and it does not have to live in one platform. It can be federated across existing systems and still behave as a single, governed source of truth. It is the operating model that keeps definitions current, assigns accountability, and stops five teams from quietly building five versions of the same answer, wherever those five versions happen to live.



The Enterprise Context Layer, the term this paper keeps returning to, is what that discipline produces: the technical and organizational expression of Enterprise Context Management.

The Enterprise Context Layer is Not a platform to buy. It is the connective structure that lets governed context reach the systems and people who need it.

A useful test for whether context is genuinely governed, or just documented, is whether these questions have one owner or five:

- What is the trusted identifier for this entity?
- Which definition applies, and who decided that?
- Is this value still reliable today?
- What action does it authorize, and for whom?
- What did the organization learn last time this judgment call was made?

If the answers scatter across five owners with no shared source, the organization is managing data. It is not managing context.

KEY TAKEAWAY

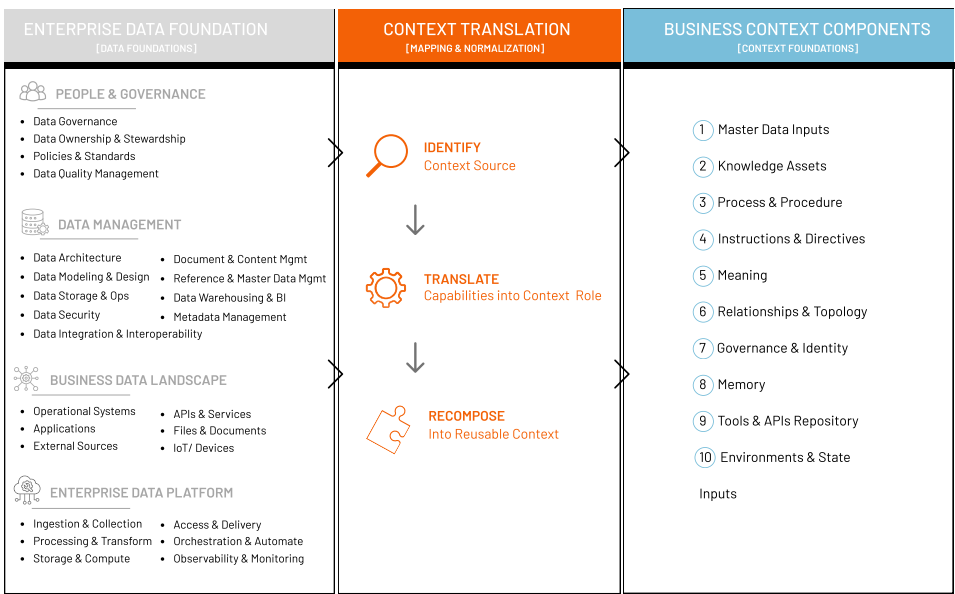
Context is the bridge between data, knowledge, action and decision. Enterprise Context Management is the discipline that keeps that bridge trustworthy, and it starts with a definition the organization actually agrees on.



2. Built for Data, Not for Context

Where does "context" actually come from?

Two decades of investment sit behind MDM, the data catalog, governance policy and the glossary. A steward can usually say, within seconds, who owns a given customer record. What that same steward cannot always say is whether the pricing rule attached to it still applies, or who approved the exception three months ago. Each discipline is mature on its own terms. None of them was built to answer that. That is what context actually is.



[Click here to enlarge](#)

This framework makes the gap visible: classical data management capabilities on one side, business context components on the other, and a translation step in between that most organizations have never formalized.

That translation follows the same logic every time.

1. Identify where a context source actually sits, a catalog entry, a governance policy, an unwritten exception a steward carries in their head.
2. Translate what that source does into a context role: meaning, identity, a rule, an ownership claim, a memory of a past decision.
3. Recompose these roles into components that analytics, reporting and AI can consume directly, instead of reconstructing them from scratch every time.

Skip that translation, and every new use case pays the reconstruction cost again, in the currency this paper has named: duplicated definitions, disagreeing systems, and governance debt nobody budgeted for.



KEY TAKEAWAY

The Enterprise Context Layer is not built from new technology. It is built by translating the Data Management capabilities the organisation already owns into reusable business context components, MDM into master data inputs, the glossary into meaning, governance into ownership, one deliberate translation step at a time.

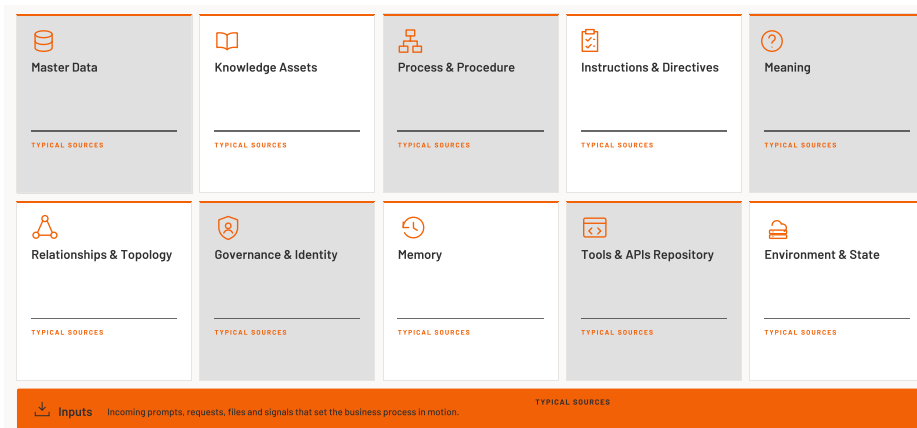
3. Ten Components, One Governed Asset

What does the layer actually need to know?

A definition tells you what enterprise context is. It doesn't tell you what to check for when it's missing, and that blind spot is where most governance efforts quietly stall.

This is not a technical architecture, and it does not prescribe a platform. It is a business description of what the Enterprise Context Layer must know, understand, constrain, access, remember and activate. Any AI, analytics or decision-making system depends on those capabilities, whether or not anyone has named them yet.

Business context breaks into ten components, governed together.



 [Click here to enlarge](#)

Each fails differently when it is missing, and the failure rarely looks like what caused it.

Meaning without Governance & Identity produces a confident answer nobody can be held accountable for. Governance & Identity without Memory relitigates the same override decision every week, because it never retained the first one.

Master Data without Memory acts on a record that was accurate three weeks ago and no longer is. This is the same convergence diagnosed earlier: retrieval, relationship, definition and ownership, meeting at the point where a decision actually gets made.

The model turns that convergence into ten things an organization can check for, rather than a description of a problem it knows it has.

KEY TAKEAWAY

Business context is not metadata with a better name. It combines ten governed components, and a layer that provides only a handful will fail in production for reasons that look unrelated but trace back to the same missing one.



4. Human Expertise Creates Tomorrow's Context

What happens to what nobody wrote down?

Project deliverables carry decision logic reasoned through once, then archived the day the project closed. Documents carry policies that were accurate when written and have quietly drifted since. People carry the rest: the tolerance granted to a key account and never written down. None of this is lost. It is simply unavailable, which is a different problem.



Knowledge Management, often reduced to document storage, becomes strategic the moment it takes on that translation. Its job is to capture what the enterprise already knows, the policies, the precedents, the tacit expertise still sitting in someone's head, before the project team disbands or the person who holds it moves on.

It looks backward, and does that job well.



Innovation Management does the job AI cannot. AI recombines what already exists, faster than any human can, but it only recombines. It cannot notice a regulation still being drafted, or price a contract of a kind never sold before. That capacity belongs to people: the decision nobody has made yet, the exception approved for the first time, the practice a team invents because no playbook covers it.

It looks forward, and creates what doesn't exist yet.

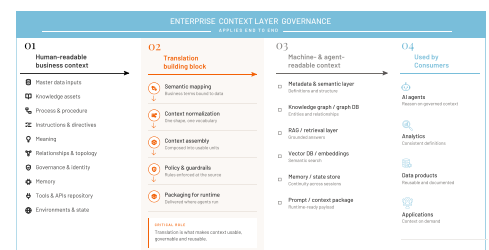


Enterprise Context Management stops either discipline's output from disappearing. Left uncaptured, new practice does not become tomorrow's context. It becomes next month's forgotten precedent, applied three different ways by three teams until nobody remembers why. It governs both directions at once: keeping existing knowledge current as processes and regulations change, and capturing new knowledge early enough to qualify it, assign an owner and fold it into the same governed layer.

It looks both ways, and lets nothing scatter.

Human-readable context depends on someone interpreting it correctly, every time. Machine-readable context is the same knowledge, structured enough that it no longer needs re-decoding.

From business context to machine context: the governed path between what people know and what agents can use.



[Click here to enlarge](#)



KEY TAKEAWAY

Without Knowledge Management, AI never learns what the enterprise already knows. Without Innovation Management, it never learns what the enterprise is about to know. Enterprise Context Management keeps both flowing into the same governed layer, instead of one being preserved and the other lost.

PART
04 | GOVERNANCE IS THE
MISSING LAYER



“Enterprise Context is a governance problem disguised as a technology problem.”

Technology can retrieve context. It cannot decide which definition is authoritative, which rule still applies, who owns it, or whether it can still be trusted today. Only governance can, and most enterprises have never assigned that mandate. They have named data owners, not decision owners. They have documented policies, not the decisions taken around them. They have built master data and metadata that make context governable, without ever connecting it to the systems that need it.

Enterprise Context Management does not start with a platform. It starts with an owner, a foundation, and a broker to move what already exists.



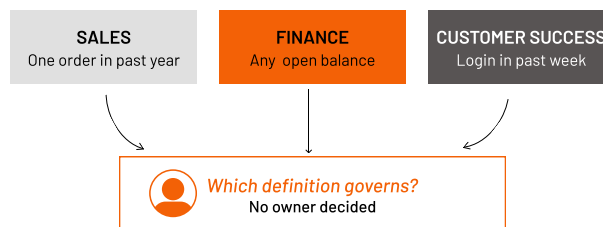
1. The Definition Needs an Owner

Who answers for the definition once it stops being obvious?

Most enterprises believe they have governed their data. What they have governed is the asset, not the decision made on top of it. That distinction is where this chapter lives.

A regional manager approved a loyalty discount override eighteen months ago. The manager has since left the role. The override was never revoked, and never re-approved either. It still executes on live transactions today, unowned, invisible to any governance framework built to track policy documents rather than judgment calls that never became one.

Definitions carry the same gap. Sales counts "active customer" as an order in the last twelve months. Finance counts an open balance. Customer success counts a login last week. Each is defensible. None was ever chosen over the others, because nobody was ever asked to choose.



Governance, as most enterprises have built it, stops at the asset. A steward owns the customer table. A domain lead owns the product catalog. Almost nobody owns the layer above: which definition is authoritative when two governed sources disagree, and who answers for that call.

For years this was settled informally, case by case, by whoever happened to know. An agent settles it too, in milliseconds, and files the result as though it had been decided.

Governance without a named owner for evolving definitions does not produce alignment. It produces plausible deniability.

This is not a call for more policy. It is a narrower mandate: someone accountable for arbitrating which definition holds, which exception still stands, which rule has quietly expired. Left unowned, that gap does not stay theoretical. It becomes the override an auditor finds, or the decision an agent makes at scale before anyone reviews it.

KEY TAKEAWAY

Enterprise Context Management starts by assigning that decision right, not documenting it further, but naming who owns it.

2. Trusted Foundations Before Trusted Context

What good is an owner without a record to own?

Naming an owner for a definition solves nothing if the record beneath it cannot be trusted. A manufacturer discovers mid-negotiation that a supplier appears under two vendor codes, one from the legacy ERP, one from an unmerged acquisition. Someone is accountable for deciding which contract terms apply. The decision right exists. What does not exist is a single, trusted version of who the supplier is. The owner arbitrates two records, not one.

Confuse a clean definition with a clean identity, and an owner ends up defending a decision built on the wrong version of the customer, the wrong version of the supplier, discovered only when a renewal, an audit, or an agent acting at scale exposes the gap.

Metadata and master data answer two different questions, and an organization can nail one while missing the other entirely. A company can document exactly what "customer" means, where the field originates, how it should be used, and still govern three unreconciled versions of the same customer as if they were separate.

LAYER	ANSWERS	WITHOUT IT
 SEMANTICS [metadata]	What does this mean, where does it come from, how should it be used?	Definitions, lineage and usage all get reinterpreted use case by use case
 MASTER DATA [identity]	Which representation of this entity should be trusted and governed?	Customers, suppliers, products or locations are duplicated and managed inconsistently

Skip this floor, and ownership becomes difficult to operationalize.

Roles are named, but there is nothing to hold them accountable to. A named owner arbitrating over three customer records instead of one is not exercising judgment. They are guessing which fragment to trust, and calling it a decision, right up until an agent makes the same guess automatically, at a scale no owner can review after the fact.

KEY TAKEAWAY

MDM and metadata are not all Enterprise Context. They are what makes Enterprise Context governable at all, the trusted floor a decision can stand on a foundation, not the decision itself.

3. The Enterprise Context Broker

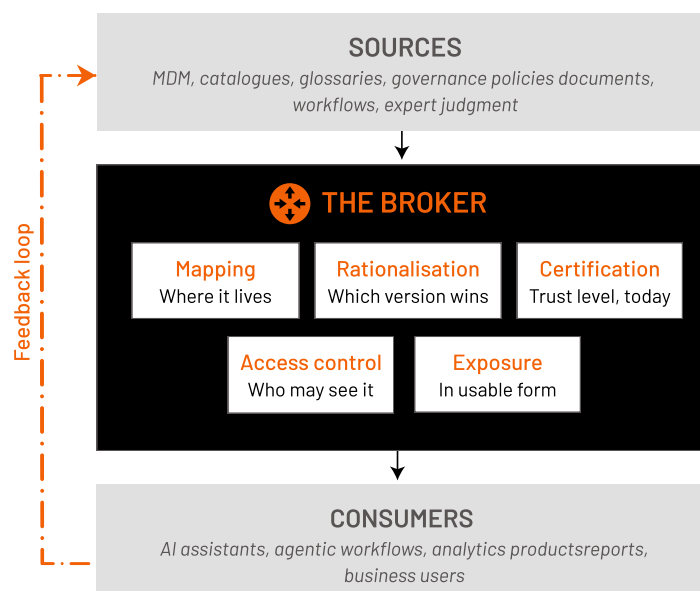
Who connects what already exists to what needs it?

On one side sit MDM systems, catalogs, glossaries, governance policies, SharePoint folders and the judgment calls stewards carry in their heads. On the other sit AI assistants, agentic workflows, analytics products, reports and business users that need exactly that material, on demand, in a form they can use. Almost nothing connects them. Each consumer builds its own bridge.

Each customer rebuilds it when the next use case needs the same answer.

The Enterprise Context Layer earns its name at exactly this point, as a broker, not a repository. It doesn't centralize every source into one platform, or replace the catalog, the MDM system or the governance policy already in place.

It sits between sources and consumers, doing the work no single discipline was ever chartered to do: locate where context lives, reconcile disagreeing versions, decide who sees what, certify how much it can be trusted, and expose the result to whichever assistant, report or agent asks for it next.



Certification is what separates a broker from a catalog.

A catalog answers a query and stops. A broker notices when an assistant keeps failing to find a definition, or when three consumers flag the same policy as outdated, and routes that signal back to the owner Chapter 1 named, before it becomes a governance gap someone discovers the hard way.

KEY TAKEAWAY

The Enterprise Context Layer is not a bigger repository. It is the broker that connects sources to consumers, governs what moves between them, and carries what it learns back to the owners accountable for keeping it right.

PART
05

WHERE ENTERPRISE CONTEXT ACTUALLY STARTS



“Nobody scales a capability by writing a policy.”

Enterprise Context Management fails the moment it tries to model the whole enterprise at once. It works when it starts small: one domain governed deeply enough to make the next one cheap, not a roadmap describing all of them. It works when strategy, enablement and continuous improvement are staffed as three separate jobs, not compressed into one project with a closing date. And it stays working only if every asset inside it carries a visible answer to one question: how much can this still be trusted, today?

The organizations ahead aren't the ones with the boldest vision.

They started narrow, and stayed disciplined.



1. Start Narrow, Govern Deep

Why does starting small feel riskier than starting everywhere?

Most Enterprise Context programs die in the same meeting: the one where someone asks for the full inventory before the first domain goes live. The belief behind that request is that breadth reduces risk. It does the opposite. A definition of "active customer" and a definition of "active supplier" left unresolved at the same time is not twice the coverage.

It is two unfinished arguments instead of one.

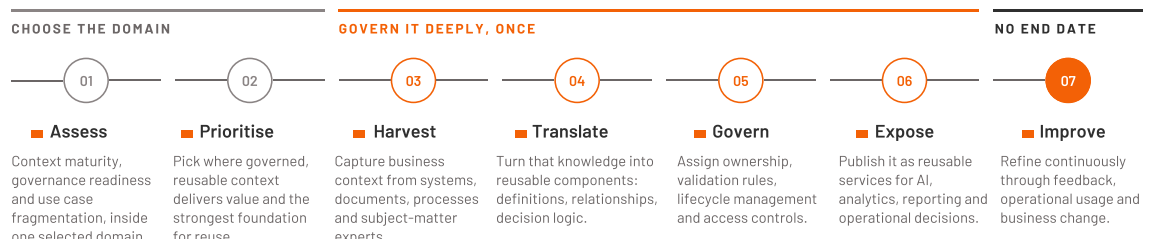
40%
of agentic AI
projects will be
cancelled by
the end of 2027

Gartner* estimates that more than 40% of agentic AI projects will be cancelled by the end of 2027, citing escalating costs, unclear business value and inadequate risk controls. The parallel with Enterprise Context is direct. An agent deployed across every domain before its underlying context is governed inherits every unresolved definition at once, and the cost of resolving them under production pressure, instead of in advance. The problem is rarely ambition. It is trying to industrialise governance before proving it.

The answer is not smaller ambition. It is sequenced ambition, and it comes down to one working principle: govern once, reuse forever.

One domain, one governance loop, one reusable component at a time. Enterprise Context becomes an enterprise capability not because it starts enterprise-wide, but because each new domain reuses the governance model, ownership principles and context services the previous one already built.

A practical Enterprise Context delivery sequence follows a deliberate path.



[Click here to enlarge](#)

The proof is never a roadmap slide. It is not even that the first use case succeeds. It is that the second use case in the same domain costs a **fraction** of what the first one did, because the context it needed was already governed, reusable, and trusted.



KEY TAKEAWAY

Enterprise Context Management doesn't scale through initial coverage. It scales the way any governed asset does: govern once, reuse forever. One domain done properly is what makes the next one fast.

2. Three Streams, Not One Roadmap

Why does a governed domain still stall six months after it goes live?

A domain gets governed, launched, and used. Six months later, a steward is overridden by a colleague who never heard the rule changed, because nobody owns updating it anymore. The domain was never at fault. The team that built it moved to the next one, and the assumption was that the work stayed finished.

That assumption is the problem.

Most organizations plan Enterprise Context Management as a single project, with a start date and a close-out date. It is not one project. It is three separate jobs, and each one runs on its own clock.

01 Strategy

Defines the vision, the governance model and the target operating model **once**, for every domain.

CADENCE



Set once. Revisited only when the enterprise itself changes.

02 Enablement

Builds and exposes reusable, governed context, **one domain at a time**. The visible delivery work.

CADENCE



Continuous, delivered domain by domain. There is always a next domain.

03 Continuous Improvement

Captures new knowledge, integrates change and keeps context **accurate** after the delivery team has left.

CADENCE



Ongoing, with no end date: the job most roadmaps forget to staff.

Confuse these three and the failure pattern is predictable.

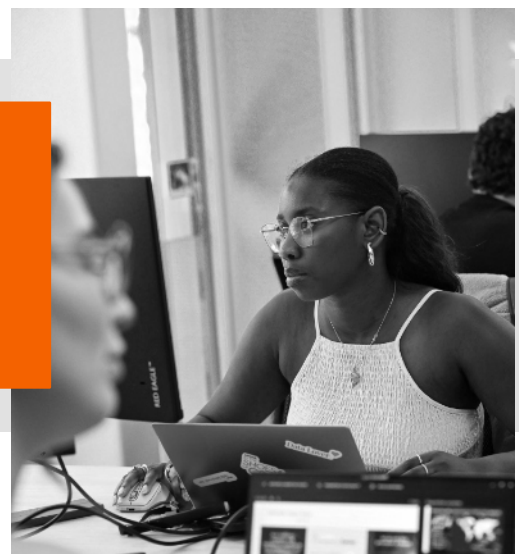
A governance model gets redesigned every time a new domain launches, because Strategy was never separated from Enablement. Or a domain goes live, its team disbands, and the context certified on day one is still being cited, unvalidated, on day four hundred, because nobody was ever assigned to watch it age.

Each stream also needs a different owner.

Strategy sits with someone accountable for the operating model across every domain. Enablement sits with the team delivering the current domain. Continuous Improvement sits with whoever is accountable for a domain once its delivery team has moved on, a role most organizations have never actually assigned.

KEY TAKEAWAY

Enterprise Context Management does not fail when a domain launches. It fails when the organization stops paying for the job that was never supposed to end.



3. Certify Once, Improve Forever

How long does "governed" stay true?

A definition gets validated, a rule gets approved, and the organization marks both settled. Context does not hold still that way. A definition stops matching the business it describes. A rule survives the process change that should have retired it. A document keeps getting cited three versions after it was superseded. Nothing fails visibly, which is the problem.

Governance without a lifecycle is a snapshot pretending to be a system.

The Broker surfaces that drift. Most organizations answer it with a single status field, and one field cannot carry three separate facts: how far a piece of context has been reviewed, where it is authorized to travel, and who may open it.

DIMENSION	WHAT IT ANSWERS	TYPICAL VALUES
01 Validation status	How much has this been reviewed, and by whom?	Draft Validated Deprecated
02 Scope	Where is it authorised for reuse?	Local Enterprise-grade
03 Access rights	Who is actually permitted to consume it?	Role-restricted Group-restricted Open

The three move independently. A regional pricing exception can be Draft, Local, and open to every team in the region that needs it today.

Scope says where a definition is authorized to travel. Access says who gets the door opened once it arrives. A single ladder forces one of them to disappear.

27%
of organizations review all AI-generated output before it is used.

The stakes are sharper for AI than for any earlier consumer of context. A stale definition in a report gets caught eventually, usually by someone who senses the number looks wrong. An agent has no such instinct, and the safety net downstream is thinner than most executives assume: McKinsey's 2025 State of AI* survey found that only 27% of organizations review all AI-generated output before it is used. Uncertified context goes in. Unreviewed decisions come out.

Certification is not a stamp that holds. It expires, and expiry sends the asset back through the same assess-to-expose sequence, to be revalidated rather than rebuilt. That work belongs to the third job named in the previous chapter: Continuous Improvement, the one with no end date and the one most organizations never staff. Certification is what makes that job visible enough to fund.



KEY TAKEAWAY

One score cannot tell a consumer whether to trust a piece of context, where it belongs, or who is allowed to open it. Certification answers all three, and it answers them on a schedule.

* Source: [McKinsey's 2025 State of AI](#)

Beyond AI

AI made the argument for this paper.
It won't be the only reason the argument matters.

AI is the most demanding consumer of context. It fails faster, more visibly, and at a greater scale than any dashboard or report ever did. But the same governed context that AI needs also improves analytics, reporting, decision intelligence, data products and human expertise. A revenue dashboard can show the correct number and still lead to the wrong decision, if nobody understands what the number measures or why it moved. A data product can be technically available and still go unused, because nobody can say who owns it or whether it can be trusted today. Human decisions fail for the same reason AI decisions fail: missing context, not missing data.

Enterprise Context wasn't built for AI.
AI is one of the reasons to build it.

Most of what it requires, the organization already owns: master data, metadata, governance policy, and the judgment calls stewards carry in their heads. What was missing was never the material. It was the discipline to connect it, govern it once, and let every consumer, human or machine, draw from the same trusted source.

During the last twenty years, organizations learned how to govern data. During the next twenty, they will learn how to govern context. AI will consume both, but it will only ever recombine what the enterprise already knows. Competitive advantage will come from mastering the second: the discipline that keeps context trustworthy, and the one that keeps new knowledge entering it.

AI is becoming a commodity. Enterprise Context will not.



Our expertise.
An offering based on 3 pillars.

DATA & AI ADVISORY

- STRATEGY
- FOUNDATION
- DATA FOR GOOD

ENABLEMENT

- DATA MANAGEMENT
- AI FACTORY
- INTELLIGENT PLATFORM
- MANAGED SERVICES

AI

BUSINESS & AI-DRIVEN SOLUTIONS

- APGARIA INTELLIGENT AGENT
- ENTERPRISE CONTENT INTELLIGENCE
- ENTERPRISE MASTER DATA ACCELERATORS

About APGAR

APGAR designs and delivers innovative data and AI solutions and supports clients with expert advisory services to ensure successful adoption and long-term value. With a team of over 230 data and AI experts, APGAR combines product development, integration, and advisory capabilities to help companies turn data into a strategic advantage.

www.apgar-group.com

