WHITE PAPER

Foundation First, Agents Second

Why Enterprise AI at Scale Requires Mastery of Two Things: What You Deploy—and What They Deploy Against

EnPraxis / Empower AI

March 2026

Executive Summary

The enterprise world is in the midst of its most consequential technology shift ever—more consequential than the cloud. The cloud was about where to store data. Agentic AI is about what runs your business.

Organizations are deploying autonomous AI agents at unprecedented speed. Gartner predicts 40% of enterprise applications will feature task-specific AI agents by end of 2026. McKinsey publicly deployed 25,000 agents across its firm. Then, in March 2026, a security researcher's autonomous agent breached McKinsey's AI platform in two hours—with no credentials, no insider access, and a decades-old SQL injection.

The industry response has been predictable: calls for better governance, tighter controls, more policies. But governance is the symptom, not the cure. And the conventional analysis misses something even more fundamental.

Core thesis:

You cannot let agents loose before you understand two things:

1. What you are deploying—the agents themselves, and the architectural foundation they require: multi-tenant isolation, protocol-level type safety, immutable governance, comprehensive observability.

2. What they are deploying against—the data, knowledge, processes, and rules that agents navigate. Even perfectly governed agents produce dangerous results when the underlying knowledge base is inconsistent, contradictory, or structurally unsound.

Solving only the first problem—agent governance—is necessary but insufficient. The second problem—knowledge integrity—is where most enterprises are blind. This paper addresses both.

This paper argues that deploying AI agents on unstructured, inconsistent enterprise knowledge is as dangerous as deploying them without governance. It presents a two-sided architecture—the Empower Foundations platform for governed agent deployment, and the EnPraxis Semantic Knowledge-Operations Fabric for structured semantic knowledge—that addresses both sides of the equation.

1. More Consequential Than the Cloud

Every prior enterprise technology wave—cloud, mobile, data lakes, analytics—shared one fundamental property: they were passive. They waited for instructions, executed them, and returned results. Security models, governance frameworks, and architectural patterns were built around this assumption.

AI agents break this model entirely. An agent does not wait for instructions. It reasons about goals, selects tools, queries databases, interprets results, and takes actions—often in multi-step chains where the output of one action becomes the input to another. When you deploy an AI agent into your enterprise, you are not adding a tool. You are introducing an autonomous actor with persistent memory, access to operational systems, and the ability to write data back into those systems.

The cloud shifted where your data lived. Agentic AI shifts what makes decisions in your business. That distinction matters because it changes the failure mode. A misconfigured cloud bucket exposes data. A misconfigured agent takes wrong actions, at scale, with compound effects, autonomously. The blast radius is not data exposure—it is operational harm.

1.1 The Scale of the Shift

- **40% of enterprise applications** will feature task-specific AI agents by end of 2026, up from less than 5% in 2025 (Gartner).
- **48% of cybersecurity professionals** identify agentic AI as the number-one attack vector—outranking deepfakes, ransomware, and supply chain compromise.
- **Over 40% of agentic AI projects** are expected to be canceled or fail to reach production by 2027 due to escalating costs and pilots that lack a path to production.
- **82% of executives feel confident** their existing policies protect against unauthorized agent actions, but only 21% have complete visibility into agent permissions, tool usage, or data access.

This confidence gap—high executive confidence paired with low operational visibility—is the defining risk of the current moment. But there is an even larger gap that almost no one is talking about: the gap between what enterprises think their knowledge base contains and what it actually says.

1.2 Why Traditional Security Models Fail

Traditional enterprise security assumes human-initiated, request-response interactions with bounded sessions and predictable access patterns. Agents violate every one of these assumptions:

- **Multi-step autonomy.** A single invocation can trigger dozens of tool calls and database queries. Each step inherits the agent's permissions, but the security model only authenticated the first step.

- **Goal hijacking.** OWASP ranks Agent Goal Hijacking as the number-one risk for agentic applications. A single malicious input can redirect an agent to perform harmful actions using its legitimate tools.
- **Cascading failures.** In multi-agent systems, a compromised agent can delegate to others, amplifying the breach. Traditional perimeter security has no model for this.
- **Knowledge-dependent reasoning.** Unlike traditional tools, agents interpret context. Their outputs depend not just on their instructions but on the quality and consistency of the knowledge they access. Inconsistent knowledge produces inconsistent decisions—and no governance framework can catch it.

2. Case Study: The McKinsey Lilli Breach

The breach of McKinsey's Lilli AI platform in March 2026 is not merely a security incident. It is a case study in what happens when an organization deploys AI at scale without foundations on either side of the equation.

2.1 What Happened

On February 28, 2026, an autonomous agent built by security startup CodeWall identified a SQL injection vulnerability in McKinsey's Lilli platform and began database enumeration. The agent required no credentials and no human involvement. Within two hours, it had gained full read/write access to the platform's database.

The scope was staggering: 46.5 million chat messages covering strategy, M&A, and client engagements; 728,000 files; 57,000 user accounts; 384,000 AI assistants; and 3.68 million RAG document chunks representing decades of proprietary McKinsey research, frameworks, and methodologies.

Most critically, Lilli's system prompts—the instructions governing AI behavior for 40,000 consultants—were stored in the same database. A malicious actor could have silently rewritten how every AI agent at the firm behaved.

2.2 Five Points of Architectural Failure

#	Failure	Impact
1	Unauthenticated APIs	22 public endpoints with no authentication; API docs publicly available
2	SQL injection	JSON keys concatenated into SQL—decades-old vulnerability undetected for 2+ years
3	No data segmentation	Single injection exposed all tenants, users, conversations, and documents
4	Prompts in user database	AI behavior rewritable through data breach—no control/data plane separation
5	No monitoring	Vulnerability existed in production for over two years without any alert

2.3 The Deeper Lesson

The governance crowd will say McKinsey needed better controls. That is certainly true. But it misses the essential question: why were the controls not there in the first place?

The answer is structural. McKinsey deployed Lilli as a rapid AI initiative—a platform built for speed and adoption, not for the architectural resilience that autonomous agents demand. When you deploy 25,000

agents on a platform without tenant isolation, without typed API contracts, without separation of control and data planes—you are not innovating. You are creating a liability that scales with every agent you add.

But there is a second lesson the industry has largely ignored. Even if McKinsey had solved every governance problem—even if the platform had been architecturally perfect—the agents would still be operating against 3.68 million RAG document chunks accumulated over decades, with no structured semantic model ensuring that the knowledge was consistent, non-contradictory, and correctly classified. Governance without knowledge integrity is like putting guardrails on a road that leads to the wrong destination.

3. The Two-Sided Problem

The conventional analysis of enterprise AI risk focuses on one side of the equation: how to control the agents. This is necessary but insufficient. The full problem has two sides, and you must solve both.

3.1 Side One: What You Are Deploying

This is the side the industry is focused on. It encompasses the agent architecture, governance model, and platform security—the structural properties that determine how agents behave, what permissions they have, and how their actions are monitored and constrained.

The requirements are well-understood even if rarely implemented: protocol-level type safety to eliminate injection, multi-tenant isolation to limit blast radius, separation of control and data planes so data breaches cannot alter agent behavior, governed agent architecture with defined roles and autonomy levels, and comprehensive observability that detects behavioral drift.

These are the failures exposed by the McKinsey breach, and they are addressed by the Empower Foundations platform through structural enforcement—not policy documents, but platform properties that cannot be bypassed.

3.2 Side Two: What They Are Deploying Against

This is the side almost no one is talking about. And it may be the more dangerous gap.

> **The knowledge integrity problem:**
> Imagine an agent navigating a complex process, system, or dataset. It may be making precisely the correct decisions given its instructions and governance constraints. But what if the underlying process is inconsistent? What if the knowledge base contains contradictions under certain

> conditions? What if the same term means different things in different contexts, and no one has reconciled them? Even the most well-governed agent will produce results that go off the reservation—because the reservation does not have the boundaries or internal consistency you think it does.

Enterprise knowledge—the accumulated processes, policies, rules, product specifications, regulatory requirements, and institutional wisdom—is rarely structured for machine consumption. It exists in documents, wikis, spreadsheets, tribal knowledge, and inconsistent databases. When enterprises build RAG systems, they typically chunk these documents, embed the chunks, and hope that retrieval will surface the right context.

This approach has a fundamental flaw: it preserves the inconsistencies, contradictions, and ambiguities of the source material. A document from 2019 says the approval process requires three signatures. A document from 2023 says it requires two. Both are in the knowledge base. Both will be retrieved. The agent will confidently cite one of them—and there is no structural mechanism to determine which is current, which applies to which context, or whether they actually conflict.

The scope of the problem

Knowledge inconsistency is not an edge case. It is the default state of enterprise information:

- **Process contradictions.** Different departments document the same process differently. Version drift creates temporal conflicts. Exceptions are documented as rules and rules are documented as exceptions.
- **Semantic ambiguity.** The same term means different things in different business units. Product names overlap. Classification systems diverge. Acronyms collide.
- **Incomplete coverage.** Critical knowledge exists only in the heads of experienced employees. Documented processes omit edge cases that practitioners handle through institutional memory.
- **Regulatory fragmentation.** Requirements differ by territory, product line, and time period. Agents need to know not just what the rule is, but which version of the rule applies to which context.

When an agent operates against unstructured, inconsistent knowledge, the result is not random error. It is confident, well-formatted, citation-backed wrong answers. The agent is doing exactly what it was told to do. The problem is that the knowledge it is working with does not mean what the organization thinks it means.

4. The Architecture: Solving Both Sides

Addressing the two-sided problem requires two interlocking architectural layers. The first governs the agents. The second structures the knowledge. Neither is sufficient alone.

4.1 Side One: The Governed Agent Foundation

The Empower Foundations platform addresses the agent governance side through five structural properties—enforced by the platform itself, not by policy:

Requirement	Empower Foundations Implementation
Protocol-level type safety	All APIs defined in Protocol Buffers with gRPC-TLS and Connect-RPC transport. Data access generated from .empower DSL models via MDE codegen—no hand-written queries. Parameterized filter DSL at framework level eliminates injection by construction.
Multi-tenant isolation	ThreadLocal TenantContext propagated through every layer. Per-tenant databases, search indices, and document storage. Per-tenant secrets in HashiCorp Vault. Data classification labels (PUBLIC through RESTRICTED) at the field level.
Control/data plane separation	Agent constitutions, prompt fragments, and governance rules deployed as versioned application artifacts (control plane). User data flows through the SEA pipeline (data plane). No shared storage between planes.
Governed agent architecture	Agentic Kernel with agent identity (role + autonomy level L0–L4), Constitution Engine with immutable governance, Guardrail Engine with monotonic tightening, and supervisor/quality-gate topology.
Observability	Five-module stack (Micrometer + OpenTelemetry + structured logging + Axon + Spring Boot). MCP Audit Filter for every tool call. LlmCallStore for every model invocation. ConversationGuard with behavioral drift detection.

These are not configurable options. They are structural properties of the platform. An application built on Empower Foundations inherits multi-tenant isolation, typed API contracts, and observability automatically. There is no deployment path that bypasses them.

4.2 Side Two: The Semantic Knowledge-Operations Fabric

The EnPraxis Semantic Knowledge-Operations Fabric addresses the knowledge integrity side—ensuring that the data, processes, and rules agents operate against are structured, consistent, and semantically grounded.

This is the architectural layer that distinguishes EnPraxis from platforms that focus exclusively on agent governance. It recognizes that in the age of AI, your company's knowledge, processes, and institutional

wisdom must be structured in a robust and consistent semantic framework that both enables understanding and supports action.

Polyglot knowledge architecture

Enterprise knowledge is not monolithic, and no single data model can represent it faithfully. The EnPraxis Fabric implements a unified polyglot knowledge store with four interlocking engines, each optimized for a different dimension of knowledge:

- **ArcadeDB: Knowledge Graph (Store of Record).** Multi-model graph database with 16+ semantic vertex types (Device, Indication, Procedure, Warning, Contraindication, Precaution, AdverseEvent, ClinicalStudy, RegulatoryApproval, and more). Typed edges define explicit relationships. This is the structural backbone—it enforces that entities, relationships, and hierarchies are explicitly modeled, not inferred from text.
- **Apache Jena TDB2: Ontology and Semantic Reasoning.** Three-tier RDF graph structure: TBox (ontology/schema with OWL classes and constraints), ABox (instance assertions as semantic triples), and SKOS vocabulary (preferred labels, broader/narrower/related relationships, synonyms). SPARQL enables semantic queries that traverse concept hierarchies, not just text matches.
- **OpenSearch: Lexical Search.** Content-type stratified indices (claims, glossary, instances, types) with BM25 scoring. Provides fast keyword-based retrieval across the structured knowledge base.
- **Milvus: Vector Embeddings.** Cosine similarity search on dense embeddings for semantic similarity retrieval. Critically, Milvus stores only embeddings and metadata—full text is always resolved from the Knowledge Graph, ensuring a single source of truth.

The key architectural decision is that ArcadeDB—the Knowledge Graph—is the store of record. Vector search and keyword search are retrieval accelerators, but they always resolve back to the graph. This means there is one authoritative representation of every entity, relationship, and fact, not four potentially inconsistent copies.

Semantic extraction with provenance

Raw documents are not simply chunked and embedded. They are decomposed into typed semantic elements—paragraphs, section headers, tables, list items, captions—each carrying full provenance: document ID, section path (hierarchical location), page number, bounding box coordinates from the original document, and a Docling self-reference for element identity.

This means every fact the system returns can be traced to a specific element on a specific page of a specific document. There is no ambiguous attribution. The provenance chain is structural, not inferred.

Ontology-driven grounding and expansion

When an agent queries the knowledge fabric, it does not perform a simple text search. The OntologyExpansionAgent grounds query entities against the Knowledge Graph and expands vocabulary through SPARQL/SKOS traversal—finding synonyms, broader concepts, narrower specializations, and related terms that the user may not have used. This ensures that an agent searching for a concept finds all relevant knowledge, regardless of which terminology was used in the source documents.

Knowledge integrity validation

The Fabric includes structural validation mechanisms that have no equivalent in conventional RAG systems:

- **Schema conformance.** Validates that relationship types declared in the ontology have actual instance records. Identifies gaps where knowledge was expected but not extracted.
- **Cross-store reconciliation.** Checks alignment between the Knowledge Graph, search indices, vector collections, and semantic assertions. Flags mismatches that indicate incomplete processing pipelines.
- **Ontology coverage analysis.** Per-document-type gap detection compares extracted entity classes against expected classes. Calculates coverage percentage and identifies missing semantic types.
- **Evidence-backed claim verification.** TrueRead verifies claims as VALIDATED, FABRICATED, UNVERIFIABLE, or PARTIAL. Epistemic Shield classifies each output element by its evidentiary status: VALIDATED, EXTRACTED, INFERRED, or UNCERTAIN.

These mechanisms address the knowledge consistency problem directly. They do not rely on agents to detect contradictions at query time. They ensure the knowledge base is structurally sound before agents ever access it.

5. Why You Need Both Sides

The two sides of the equation are not independent. They interact in ways that make solving only one side dangerous.

Governance without knowledge integrity

An enterprise deploys agents with perfect governance: multi-tenant isolation, guardrails, observability, supervision. The agents query a RAG system built on unstructured document chunks. One document says the inspection interval is 12 months. Another says 6 months. A third references a superseded standard. The agent retrieves all three, selects the most frequently cited answer, and confidently recommends a 12-month interval with citations. The governance system reports: all guardrails passed,

all actions within scope, all responses logged. The answer is wrong, and the governance system cannot know it is wrong, because it governs the agent's behavior, not the knowledge's consistency.

Knowledge integrity without governance

An enterprise builds a structured semantic knowledge fabric with validated ontologies, reconciled entities, and provenance tracking. They deploy agents against this high-quality knowledge base without proper governance. An attacker injects a prompt through a document the agent retrieves. The knowledge is correct, but the agent's behavior is hijacked—because there are no guardrails, no constitution enforcement, no behavioral monitoring. The structured knowledge could not protect against a governance failure.

> **The two-sided principle:** You must govern what you deploy (the agents) AND structure what they deploy against (the knowledge). Solving one without the other creates a false sense of security. The Empower Foundations platform solves the first. The EnPraxis Semantic Knowledge-Operations Fabric solves the second. Together, they form the complete foundation that enterprise AI at scale demands.

6. Implications for Enterprise Leaders

The McKinsey breach and the broader industry trajectory should prompt every enterprise leader deploying AI agents to ask five questions:

1. What is the blast radius of your next vulnerability?

If a single injection point can reach all tenants, all users, and all data, your blast radius is your entire enterprise. Multi-tenant isolation is the single most important architectural decision for limiting blast radius.

2. Where do your agent behavior definitions live?

If system prompts and governance rules are stored in a database that application code reads at runtime, they are in the blast radius of any data breach. Separation of control and data planes is not a best practice—it is a requirement.

3. Can your agents be observed?

If you cannot trace every step of an agent's execution—every tool call, model query, and governance decision—you do not have observability. The distinction matters because agents fail through gradual behavioral drift that traditional logging cannot detect.

4. Is your knowledge base consistent?

If your RAG system retrieves conflicting information from different source documents and your agents cannot distinguish current from superseded, authoritative from provisional, or applicable from inapplicable—your governance controls are protecting a flawed foundation.

5. Can you validate your knowledge before agents access it?

If you cannot measure the coverage, consistency, and semantic integrity of your knowledge base independently of agent queries, you are relying on agents to discover knowledge problems at runtime—the most expensive and dangerous time to find them.

6.1 The Governance Paradox

Forrester predicts 60% of Fortune 100 companies will appoint a head of AI governance in 2026. Gartner projects the AI governance platform market will reach billions. These are necessary responses to a real problem.

But governance without architecture is policy without enforcement. And governance without knowledge integrity is enforcement without foundation. An AI governance framework that does not address the quality, consistency, and structural soundness of the knowledge agents operate against is solving half the problem—and the easier half at that.

7. Conclusion

The enterprise AI industry is at an inflection point. The McKinsey breach has made visible what has been true since the first agent was deployed into a production system: autonomous AI agents demand a foundation that prior enterprise technologies did not require.

But the foundation is not just governance. It is two things:

> **Foundation first, agents second—on both sides:**
>
> **1. Master what you deploy.** Build the multi-tenant isolation, typed API contracts, immutable governance, and comprehensive observability that make the McKinsey breach pattern structurally impossible. This is the Empower Foundations platform.
>
> **2. Master what they deploy against.** Structure your enterprise knowledge in a semantic framework that ensures consistency, tracks provenance, validates coverage, and detects contradictions before agents encounter them. This is the EnPraxis Semantic Knowledge-Operations Fabric.

> **3. Neither side is optional.** Governance without knowledge integrity produces well-monitored wrong answers. Knowledge integrity without governance produces correct answers through exploitable channels. Both must be solved, and solved together.

The firms that will deploy AI agents safely and effectively at enterprise scale are not the ones with the most comprehensive governance policies or the largest RAG systems. They are the ones that built the foundations—on both sides—before the first agent was deployed.

Because when you scale agents on a foundation that governs their behavior and structures their knowledge, you are not multiplying exposure. You are multiplying capability with confidence. And that is the difference between enterprise AI as a liability and enterprise AI as a genuine strategic advantage.

Prepared by EnPraxis / Empower AI | March 2026 | enpraxis.ai