ContextOne: A Context and Control Architecture for Enterprise AI

Shaun Laurens[1], **Mark Sykes**[1], **Fergus Keenan**[1]

[1]AI One

Agentic AI has moved from research curiosity to enterprise priority, yet production deployment has stalled. The frameworks that emerged from the conversational AI era treat context as a single, ever-expanding payload and embed governance in prompts rather than architecture. The result is a class of system that is powerful in demonstration and unreliable in production, failing on the dimensions that regulated enterprises require most: deterministic compliance, auditable reasoning, and economic scalability.

We decided a fundamental redesign was necessary. Over the past 18 months, we built ContextOne from the ground up as a composable context and control layer for enterprise-grade agentic workflows. ContextOne is a stateless, event-driven runtime: modular, governance-native, and capable of sustaining 100,000+ concurrent agentic workloads across a small number of nodes. Agents materialize on demand, execute their iteration, and yield resources until the next event, with state restored from a shared cache in single-digit milliseconds. It is deployed in production across regulated enterprise environments today, processing complex, multi-step workflows against live enterprise systems under full audit and access control.

In this paper, we describe the architecture of ContextOne and highlight two primary contributions. First, the Agent Harness: a novel stateless runtime that separates agent reasoning from execution to deliver accuracy and concurrency at scale. Second, a suite of deterministic enterprise controls — spanning deterministic query enforcement at the data layer, formal Z3 SMT constraint validation, cryptographic audit, and end-to-end data lineage with propagating access controls — that provide verifiable trust for mission-critical deployment.

On Tau Bench 2, an established benchmark for agentic task completion, ContextOne achieves greater-than-3x improvement over baseline, moving task completion from approximately 30% to 95%, driven entirely by architectural choices rather than prompt optimisation. An agent operating at 30% task completion requires constant human supervision; at 90%+, it can be delegated to real business processes, with human oversight reserved for genuine exceptions.

Date: March 31, 2026
Correspondence: shaun@ai.one, mark@ai.one, fergus@ai.one

1 Introduction

Large Language Models have fundamentally changed what enterprise automation can look like, shifting the possibility space from rigid, pre-programmed workflows to dynamic, reasoning-driven processes. The agentic frameworks that emerged from this shift — LangChain, CrewAI, and their peers — have proven capable for orchestrating dialogue and simple tool use LangChain (2024); Aguilar (2023). As enterprises have attempted to deploy these systems for mission-critical processes, however, a consistent set of architectural limitations has emerged as the primary barrier to production adoption.

These frameworks treat context as a single, ever-expanding window. The consequences are well-documented:

- **Lack of Granular Control:** All context is treated equally, making it impossible to prioritize critical

information, protect immutable data, or apply layer-specific optimizations.

- **Scalability and Cost:** As conversational history and tool outputs accumulate, the context window becomes bloated, increasing costs and degrading model focus Ge et al. (2023).
- **Governance and Auditability Gaps:** Without structured context, enforcing business rules, applying security policies, and maintaining a tamper-evident audit trail become intractable problems.
- **Inherent Non-Determinism:** The probabilistic nature of LLMs makes them unreliable for processes requiring predictable, verifiable outcomes Atil et al. (2024).

The industry has largely responded to these architectural problems with prompt-level solutions. The analogy to data warehousing before the cloud era is instructive: on-premise, shared-nothing systems were poorly suited to the elasticity and scale of the cloud, and no amount of tuning resolved what was fundamentally an architectural mismatch. A new architecture — one that separated storage from compute — was required Dageville et al. (2016). The shift from conversational AI to production-grade autonomous AI presents the same kind of problem.

1.1 The Fourth Paradigm: The Enterprise Agent Runtime

Today's agent-like systems fall into three broad categories: visual workflow builders that execute predefined sequences, often without reasoning; agent frameworks such as CrewAI and LangChain that provide reasoning loops but embed governance in prompts rather than structure; and CLI tools such as Claude Code and Codex that are powerful but provide no centralized policy enforcement, data lineage, or audit. ContextOne occupies a fourth category: the Enterprise Agent Runtime. It delivers the autonomous reasoning of CLI tools with the architectural guarantees, centralized governance, and full auditability required for mission-critical deployment.

The remainder of this paper is structured as follows. Section 2 details the five pillars of the ContextOne architecture. Section 3 explores how the Agent Harness enables accuracy and scale. Section 4 details the enterprise controls that ensure safety and reliability. Section 5 presents our performance evaluation results. Section 6 discusses related work, and Section 7 concludes with lessons learned and future directions.

2 The ContextOne Architecture

ContextOne is designed as a service-oriented architecture of independently scalable, fault-tolerant services, organized as a composable stack of five core pillars. Each pillar is exposed via well-defined APIs and can be deployed selectively. Enterprises adopt the capabilities they need without carrying the weight of what they don't. This API-first, modular design means ContextOne functions as an open integration layer rather than a monolithic system, sitting cleanly alongside existing enterprise infrastructure rather than replacing it.

2.1 The AI Data Fabric: A Foundation for Enterprise Understanding

At the data foundation of the stack is the AI Data Fabric — a powerful interpretation engine that connects to and understands enterprise data sources, both structured and unstructured, without requiring data migration. Gateways in this layer are not just pipes; they are security checkpoints that abstract connection details and enforce security policies before any data is returned to the agent.

The Schema Interpreter analyzes the metadata of connected sources, automatically extracting, normalizing, and organizing the structural information agents need to interact with those systems safely and correctly. This removes a significant manual onboarding burden and eliminates a common class of integration errors that arise when agents operate against poorly understood schemas.

Where enterprise deployments require entity resolution across disparate systems, the Fabric provides a five-stage entity-matching pipeline to resolve entities across disparate systems: LLM-based standardization normalizes records from heterogeneous sources without hand-crafted parsing rules; deterministic matching links records sharing hard identifiers; fuzzy matching extends the candidate set using phonetic encoding and string distance metrics; semantic embedding similarity resolves entities that are lexically dissimilar but conceptually equivalent; and an LLM adjudicator evaluates remaining candidates in context, producing a

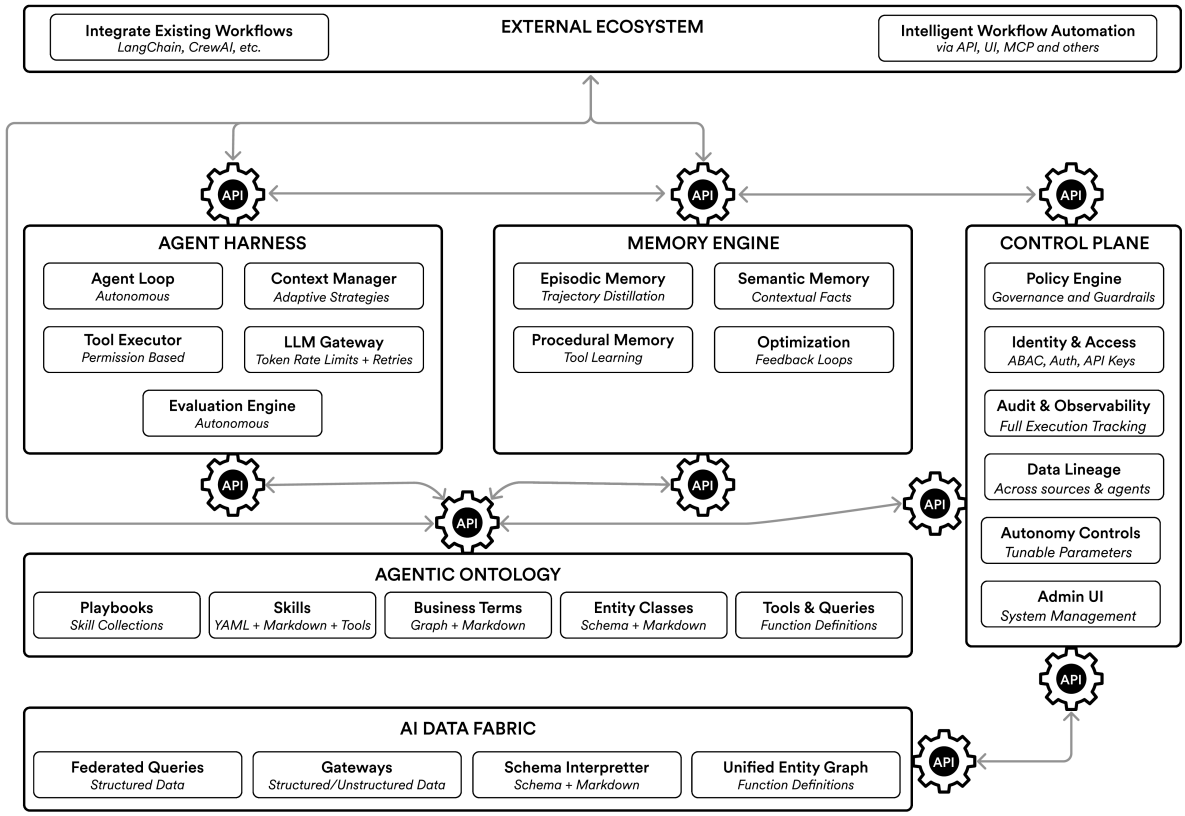

Figure 1 The ContextOne context and control architecture.

confidence-scored decision with a human-readable rationale. The result is a Unified Entity Graph that agents query as a coherent, semantically consistent view of the enterprise.

2.2 The Agentic Ontology: A Map of Enterprise Meaning

Above the Fabric lies the Agentic Ontology — a governed repository of enterprise knowledge comprising playbooks, skills, business terms, and entity classes that together define an enterprise's operational reality. Any existing enterprise application or agent can draw on it directly, providing a shared foundation for enterprise context rather than requiring each team to reconstruct it from scratch.

Tools & Queries are explicitly part of the ontology, defining what can be done distinct from the runtime execution of those tools. Named Queries are a specific class of query tool within the ontology that agents invoke by name rather than constructing queries against live systems at runtime. The query logic is fixed and versioned; the agent supplies parameters, not structure. This shifts query execution from probabilistic to deterministic at the ontology level and the agent can only invoke what has been defined, reviewed, and approved.

Critically, the ontology isn't hand-built, and it doesn't require manual curation to stay current. It is hydrated automatically from the AI Data Fabric's interpretation of existing enterprise systems, alongside any additional structured, unstructured, or semi-structured data sources, such as existing SOPs and policies, and it continues to grow as workflows run and operational feedback is received. The result is a dynamic ontology that evolves with the business: new data sources, reorganized teams, updated policies, and changing regulatory requirements are absorbed and reflected without human intervention.

This separation of concerns is deliberate. The AI Data Fabric creates a coherent, unified view of the enterprise's data estate; the Agentic Ontology enriches that view with business logic, playbooks, and skills to produce the

agent-ready context that drives intelligent action.

2.3 The Agent Harness: A Runtime Container for Agents

A key architectural innovation of ContextOne is the formal separation of an agent's reasoning logic from its runtime environment. The Agent Harness is the stable, managed runtime container where an agent executes, while the Agent Loop — the agent's internal, recursive heartbeat — provides a series of governed pipelines which together constrain probabilistic reasoning within deterministic, auditable guardrails. The loop reasons; the harness governs.

This separation is crucial for providing a reliable, auditable environment for inherently non-deterministic agents. By isolating the execution environment, we can enforce security policies, manage resources, and provide a consistent, observable runtime, regardless of the specific task the agent is performing. The Evaluation Engine (see Section 4) is explicitly placed inside the harness, reflecting the "Judge Prompt" concept where the system deterministically validates steps or requests Human-in-the-Loop (HITL) intervention before proceeding.

2.4 The Memory Engine: Three Pipelines for Continuous Learning

The Memory Engine is where ContextOne learns. We required an approach beyond generic memory systems, which use a combination of vector databases, knowledge graphs, and other techniques to store and retrieve information. The result is a more precise and powerful architecture inspired by human cognition. The Memory Engine is not a monolithic block, but a dynamic, asynchronous system of feedback loops that continuously refines the agent's understanding. It consists of three distinct pipelines, each specialized for a different class of learned knowledge, and a continuous optimization process that refines all three through feedback loops.

- **Episodic Memory (Trajectory Distillation)**: This pipeline remembers "what happened" in a specific sequence. It reviews the agent's full execution trace, clusters similar activities, and identifies "golden trajectories" that represent best practices.
- **Procedural Memory (Tool Learning)**: This pipeline remembers "how" to use a tool. It analyzes tool outcomes and updates the prompt snippets that guide future tool use, improving the agent's ability to use its tools effectively over time.
- **Semantic Memory (Contextual Facts)**: This pipeline remembers "facts" about the user or business context. It extracts and stores contextual facts from the agent's interactions, building a rich, persistent model of the enterprise's operational reality.

These three asynchronous processes run in parallel with the main agent loop, ensuring that the system is always learning, improving, and adapting to new information.

2.5 The Control Plane: An Open Enterprise-Grade Governance Layer

The Control Plane is the governance layer of ContextOne, applying controls as a series of layered, independently enforceable mechanisms across the full lifecycle of an agent's execution. Rather than embedding governance in prompts or orchestration logic, ContextOne enforces it structurally at the architecture level, not the LLM level.

A core security invariant is that the Agent Harness never holds credentials. All API keys, secrets, and connection credentials are managed exclusively at the gateway layer and never propagate into the agent's context window. A fully compromised Agent Loop cannot leak what it never possessed.

Access control is enforced via Attribute-Based Access Control (ABAC) through Open Policy Agent, with the agent's and user's attributes composed and evaluated against every resource request at runtime. Data lineage is captured automatically at the point of access and propagates across agent boundaries and into output artifacts, with access controls travelling with the data at every stage.

All activity is recorded through a cryptographically signed audit ledger using W3C Decentralised Identifiers, verifiable with standard tooling and no dependency on ContextOne itself.

3 Achieving Accuracy and Scale with the Agent Harness

The leap from a promising pilot to a production system that spans the enterprise hinges on an architecture that delivers both reliable accuracy and scalable concurrency. This is the central design mandate of the Agent Harness. By separating the agent's reasoning from its execution environment, the harness provides an auditable, highly concurrent runtime.

Two key architectural choices enable this: a stateless, event-driven execution model and a deterministic, seven-stage control loop. Together, they transform the agent from a persistent, memory-resident process into a lightweight, serverless-like function that materializes only when work is required.

3.1 The Stateless Execution Model: Built for Concurrency

A key architectural choice distinguishes this approach from conventional frameworks: agents that are not actively performing work are not memory-resident. An agent exists as running code only when it has work to do. Between iterations, its state is externalized and when execution resumes, ContextOne retrieves the agent's context from a shared cache, achieving resume latency typically in the single-digit milliseconds. This stateless, event-driven design allows ContextOne to sustain a large population of concurrent agents across a small number of nodes, demonstrated at scale to hundreds of thousands, dramatically reducing the infrastructure cost of enterprise-scale deployments. The result is an execution model closer to a serverless function than a traditional long-running daemon, in which agents materialize on demand to handle a task and then disappear.

Because idle agents consume zero compute resources, the cost curve is driven by active work, not by the number of agents deployed. An enterprise can deploy a specialized agent for every employee, every workflow, or every customer without incurring a prohibitive baseline infrastructure cost.

3.2 The Governed Agent Loop: Building Accurate, Auditable Work

The Agent Loop is a multi-stage governed pipeline that constrains probabilistic reasoning within deterministic, auditable guardrails through seven stages of governance applied at every step.

The full loop decomposes into the following stages:

1. **Prompt Assembly:** The harness finalizes the system prompt and tool definitions in an LLM-agnostic canonical representation, independent of any specific model provider. This representation is translated into the appropriate vendor-specific format at invocation time, preserving the separation between reasoning logic and the underlying model infrastructure.
2. **Context Management:** A configurable pipeline of transformations manipulates the data flowing into the LLM, optimizing each content class independently. This includes strategies like token-efficient reformatting and result offloading.
3. **LLM Invocation:** The assembled prompt is dispatched to the configured model, with an optional waterfall strategy for transparent failover to alternative models.
4. **Result Validation:** The LLM's response is deterministically validated against security policies and exception rules before any further processing.
5. **Tool Execution:** A three-phase pipeline applies pre-execution controls, dispatches the call through a load-aware scheduler, and applies post-execution controls.
6. **Result Collection and Context Gating:** Large tool results are not injected directly back into the context window. Instead, the agent receives a pointer and a set of tools to process the data efficiently, preventing context bloat and maintaining focus.
7. **Control Condition Evaluation:** The harness verifies that cost and iteration thresholds have not been exceeded. When limits are reached, the harness either escalates to a human for permission to continue or gracefully halts execution after persisting the agent's current state.

This multi-stage process enforces security, observability, and compliance at every stage of every agent iteration, creating a tamper-evident audit trail by default. The combination of context gating and the approximately twenty built-in context management strategies means agents operate with smaller, cleaner context windows, reducing both token cost and hallucination risk as task complexity grows.

4 Controls for the Agentic Era

In an era where AI agents can act on an enterprise's behalf, the ability to govern, audit, and deterministically control their behavior is a prerequisite for adoption. ContextOne enforces safety structurally as a property of the architecture rather than a consequence of prompt design. This is achieved through a suite of integrated controls that provide defense-in-depth for agentic workflows, from real-time behavioral validation to a tamper-evident audit trail.

ContextOne enforces controls structurally, across every stage of the agent's execution lifecycle: from the moment an LLM response is received, through tool execution, to final verification of a completed task. The sections that follow detail the four control surfaces that constitute ContextOne's defense-in-depth architecture: a three-tiered evaluation framework, centralized identity and access management, deterministic constraint enforcement, and end-to-end data lineage with propagating access controls.

4.1 A Three-Tiered Evaluation Framework for Building Trust

A critical requirement for enterprise deployment is verifiable correctness. ContextOne's three-tiered evaluation framework operates at distinct points in the agent's execution lifecycle, each addressing a different dimension of correctness — from enforcing the sequencing of individual tool calls, through principle-based validation of agent decisions, to independent verification of task outcomes. Together, these tiers create a defense-in-depth strategy that catches errors at multiple stages without requiring human intervention at each one.

4.1.1 Tier 1: Instant Eval (Behavioral Guardrails)

Instant Eval enforces deterministic, real-time constraints on the sequencing and preconditions of tool calls. Before an agent can execute a tool call, the Instant Eval engine checks whether all required preconditions have been satisfied. If a check fails, the agent receives clear, actionable feedback — "you must read the file before updating it" — rather than a silent rejection, enabling the Agent Loop to self-correct without human intervention. This tier is particularly effective for tightly defined agentic flows or interdependent API calls where the correct sequence of operations is known in advance.

Example: An airline booking agent attempts to reserve a seat without first checking availability. Instant Eval detects that the check_seat_availability tool has not been called — a required precondition — and blocks execution. The agent receives specific feedback instructing it to verify availability first, after which it can proceed with the reservation.

4.1.2 Tier 2: LLM Judge (Principle-Based Validation)

Where deterministic rules are insufficient, the LLM Judge provides principle-based validation at designated checkpoints, evaluating whether an agent's actions align with higher-level objectives and constraints. The Judge operates with a filtered view of conversation history — receiving only the information relevant to its assessment — allowing validation intensity to be calibrated to the stakes of the decision without consuming unnecessary context. It returns a binary decision alongside detailed feedback, and is independently configurable per agent and per checkpoint.

Example: A loan approval agent proposes approving a $50,000 loan to a customer with a credit score of 620. The LLM Judge, configured with the firm's risk-management principles, determines that the loan amount exceeds what is permissible for that credit profile. It rejects the action and returns specific feedback, for example, that the amount should be reduced or additional collateral required, allowing the agent to revise its recommendation rather than fail silently.

4.1.3 Tier 3: Agentic Grader (Task Completion Validation)

The Grader activates when an agent declares its task complete, verifying that the intended outcome was actually achieved in the underlying systems rather than simply claimed. It supports two complementary verification strategies. For tasks where verification can be reduced to a deterministic check, the Grader invokes tools directly — Python scripts, database queries, or custom gateway tools — against the target system. For tasks requiring more nuanced assessment, it dispatches sub-agents configured with dedicated introspection tools, allowing them to examine the primary agent's full activity history and confirm that recorded results are consistent with the expected downstream state.

The outputs of both strategies feed into a dedicated LLM Judge that renders a binary decision: task completion is confirmed, or specific deficiencies are fed back to the Agent Loop for correction before the agent may attempt to declare completion again. Agents do not simply declare victory; they must demonstrate it, evaluated by an independent process they cannot influence. Grading outcomes and evaluated trajectories additionally serve as inputs to ContextOne's continuous learning pipelines, providing the raw material from which agent behavior improves over time.

Example: An agent claims to have updated a customer's seating assignment. Rather than accepting this at face value, the Grader executes a direct database query against the airline's reservations system to confirm the change is present. If it is not, the Grader identifies the specific discrepancy and returns it to the Agent Loop and the agent must address the failure and re-submit for verification before the task can be marked complete.

4.2 The Control Plane: Identity, Access, and Audit

The Control Plane's primary mandate is to manage identity, control access, and maintain a complete, unalterable record of every action taken by any agent across the system.

The Agent Harness never holds credentials. The Control Plane manages Identity & Access, ensuring that even if an agent is compromised, it cannot leak keys because it never possessed them. This is achieved through a governed gateway architecture, where all credentials, API keys, and secrets are managed exclusively at the gateway layer and never propagate into the agent's context window. Security is enforced via Attribute-Based Access Control (ABAC), which composes the user's attributes with the agent's attributes to make real-time decisions about data access before any tool is invoked or any data is returned.

Furthermore, the Control Plane provides deep observability. The cryptographic audit module provides a signed, tamper-evident ledger of every thought and action, crucial for enterprise compliance. This isn't a simple log file; it's a cryptographically verifiable chain of evidence that provides full data lineage, tracking what data was accessed, by which agent, for what purpose, and how it was transformed. This gives regulated industries the provenance guarantees they require without adding manual tracking overhead.

4.3 Deterministic Validation: Formal Constraint Enforcement

For business rules that must never be violated, even the most sophisticated LLM-based evaluation is insufficient. ContextOne incorporates a deterministic validation layer for these scenarios. This layer uses the Z3 Satisfiability Modulo Theories (SMT) solver to enforce logical constraints expressed as formal rules. When an agent attempts to execute a tool call that could violate one of these rules, the proposed action is translated into a logical statement and checked against the rule set. If the action violates a rule, the execution is deterministically blocked.

Example: An agent attempts to call the approve_loan tool with parameters indicating a credit score of 620. The Z3 solver evaluates the rule $\texttt{credit\_score} < 650 \rightarrow \texttt{loan\_denied} = \texttt{true}$ *and determines that the proposed action violates this constraint. The tool call is blocked.*

This deterministic layer provides a mathematical guarantee of compliance, eliminating the possibility of LLM hallucinations or reasoning errors leading to rule violations in production.

4.4 Data Lineage

ContextOne captures data lineage automatically at the point of access and propagates it across agent boundaries and into output artifacts, providing an unbroken provenance chain from source system to final deliverable. Access controls travel with lineage and sensitivity attributes are enforced on every downstream consumer of the data, however many agents have transformed it, with no requirement for manual re-application at each step.

5 Performance Evaluation

Evaluating enterprise AI systems requires balancing theoretical rigor with practical applicability. Standardized benchmarks provide controlled environments for measuring capability, but the metrics that matter most to enterprises — governance compliance, token economics, and the ability to complete complex, multi-step tasks autonomously — are rarely what benchmarks are designed to test.

The benchmark that comes closest to capturing what enterprise agentic AI actually needs to do is Tau Bench 2. Unlike benchmarks that test isolated question-answering or single-tool retrieval, Tau Bench 2 evaluates an agent's ability to complete end-to-end tasks autonomously across complex, multi-turn workflows — tasks that require sequencing tool calls correctly, handling ambiguity, recovering from errors, and ultimately producing a verifiable outcome in an underlying system. This is the class of problem that enterprise agentic AI must solve. An agent that can answer a question accurately is useful; an agent that can execute a business process reliably and autonomously, from initiation to verified completion, is transformative. Tau Bench 2 measures the latter.

5.1 Performance Benchmarking: Tau Bench 2

ContextOne was evaluated against Tau Bench 2 across two domains: airline and telecoms. The results demonstrate substantial performance improvements over the baseline in both.

Domain	Task Set	Baseline (Claude Opus 4.5)	ContextOne	Improvement
Telecoms	Full (2,400 tasks)	∼30%	95%	>3x
Airline	Restricted (verified solvable)	∼40%	78%	∼2x
Airline	Full (all tasks)	∼25%	52%	∼2x

Table 1 Tau Bench 2 Results: Baseline vs. ContextOne.

In the airline domain, ContextOne achieved a 78% success rate on a restricted subset of manually verified solvable tasks, and 52% across the full task set. In telecoms, ContextOne achieved a 95% success rate across approximately 1,146 completed tasks. These results were achieved using the same prompts and input data as the baseline. Performance gains stem from ContextOne's architecture rather than benchmark-specific prompt optimisation. The same prompts were used for both baseline and ContextOne runs without modification.

5.2 What These Results Mean in Practice

For teams evaluating agentic infrastructure, the Tau results have a direct practical implication: the architecture governing how an agent manages context, validates its own behavior, and verifies task completion has more impact on outcomes than the underlying model. Swapping Claude Opus 4.5 for a more capable model in a monolithic framework will yield incremental gains. Restructuring how context is managed, how tool calls are governed, and how task completion is verified produces the order-of-magnitude improvements reflected in the table above.

For enterprises adopting this architecture, this translates to agents that can be trusted to execute multi-step business processes autonomously, not just assist with them. The difference is not academic. An agent operating at 30% task completion requires constant human supervision and produces little net efficiency gain. An agent operating at 90%+ can be delegated real work, with human oversight reserved for genuine exceptions rather than routine error correction.

5.3 Our Tau Implementation Choices

To ensure a fair benchmark, we made a series of deliberate implementation choices. We used the exact same prompts as the baseline without modification, isolating the impact of architecture from prompt engineering. It is worth noting that the telecoms baseline prompts are notably weak — a characteristic of the unmodified Tau benchmark that external evaluators typically address through prompt customisation. We elected not to do so in order to maintain a consistent and reproducible comparison basis. The performance gains reported therefore represent what ContextOne's architecture contributes over an unmodified baseline, not a cherry-picked starting point.

Our implementation differed from the baseline in several key ways:

- The Tau Bench Bot Agent was not executed directly. Instead, the ContextOne Agent Harness executed the bot agent's logic using its own tools and prompts.
- An LLM Judge was configured to validate agent behavior before executing specific tool calls.
- Tool names were standardized to RFC 1035 for security reasons.
- A dedicated Tau Gateway process was created to implement and expose all necessary tools.
- A PostgreSQL database was used to process and store benchmark data, seeded at startup and reset between tests.
- The Tau Bench User Agent communicated via a logic-free REST bridge to ContextOne.

Critically, ContextOne was doing substantially more than the baseline during execution — operating against a real database, applying Control Plane security checks on every LLM operation, and invoking the LLM Judge before tool execution. The performance gains were achieved in a more realistic, production-like environment, not a sterile in-memory benchmark.

5.4 Benchmark Limitations

Current benchmarks have inherent limitations that the industry needs to address. Tau Bench 2 includes tasks that are not solvable given the provided data, making 100% success rates structurally impossible and complicating direct score comparisons. Binary pass/fail scoring also fails to capture partial progress, masking meaningful differences between an agent that makes no progress and one that completes 90% of a task correctly before failing at the final step.

Future benchmarks must measure the full economic and operational reality of enterprise AI: token consumption per task, the agent's ability to self-correct mid-execution, and the quality of human escalation when intervention is genuinely required. The methodology we applied here — isolating architectural impact, using a real database, and running governance controls during the benchmark — represents the kind of evaluation standard that enterprises should demand when assessing agentic infrastructure.

6 Related Work

Retrieval-Augmented Generation (RAG). RAG has emerged as the dominant paradigm for providing external knowledge to LLMs Lewis et al. (2020). The core idea is to retrieve relevant documents from a knowledge base (typically a vector database) and include them in the context. While powerful, we view RAG as a crucial component of a broader context management strategy, not the strategy itself. AI One extends the RAG paradigm by integrating retrieval into a multi-layered architecture. The AI Data Fabric provides a much richer source of information than a simple vector database, and the Agentic Ontology provides a semantic layer that enables more intelligent, context-aware retrieval.

Context Engineering and Optimization. The survey by Mei et al., which analyzes over 1,300 papers on context engineering, and Google's November 2025 whitepaper on context management for LLM-based agents Mei et al. (2025); Milam and Gulli (2025) are useful primarily as illustrations of the problem AI One was designed to solve. Both bodies of work document the growing complexity of managing context at scale, and recognize that context is not a single undifferentiated input. ContextOne builds on that direction by making the separation

operational and central to the architecture. Packing context into distinct, governed layers, each of which can be managed with its own optimization and control strategy.

The Open Semantic Interchange (OSI). The recent introduction of the Open Semantic Interchange (OSI) by Snowflake and its partners represents a significant and valuable step toward standardizing the language of business Snowflake (2025). By creating a vendor-neutral, open standard for defining semantic layer constructs like metrics and dimensions, OSI provides a much-needed vocabulary for enterprises to achieve consistency across their BI and AI tools. It addresses the critical problem of "semantic drift," in which terms such as "net revenue" or "active customer" can have different meanings across systems, leading AI to confidently produce incorrect answers. However, a shared vocabulary, while essential, is not sufficient for intelligent communication. AI cannot form insightful questions or construct well-structured answers from a dictionary of terms, dimensions and relationships alone. This is where ContextOne provides the necessary complement to a standard like OSI. While OSI provides the "what" (the standardized business terms), AI One provides the "how": Context Injection, the Memory Engine, and the Evaluation Framework. In short, while OSI is a foundational layer for defining the language of business, it falls short of enabling AI to move beyond simple Q&A towards enterprise-level problem-solving.

Deterministic Behavior and Formal Verification. The challenge of non-determinism in LLMs is well-documented Atil et al. (2024). While some approaches seek probabilistic solutions, our work aligns with research that aims to impose deterministic constraints on probabilistic systems. The "Blueprint First, Model Second" framework Qiu et al. (2025) and the AgentGuard runtime verification system Koohestani (2025) similarly advocate for structured, verifiable workflows. Our use of SMT solvers for business rule validation is a practical application of formal methods, creating a hybrid system that combines the generative power of LLMs with the logical rigor of constraint satisfaction, providing a robust solution for enterprise safety and compliance.

Ontology-driven enterprise platforms. Among existing enterprise AI platforms, Palantir's Foundry and AIP stack is one of the closest architectural analogues to ContextOne. Foundry's Ontology models real-world entities, properties, relationships, and actions as a persistent semantic layer over enterprise operations, while AIP connects large language models to ontology-mediated workflows through components such as Agent Studio and AIP Logic Palantir Technologies (2024, 2025). The result is a deeply integrated environment in which agents can consume ontology objects, invoke governed actions, and operate with strong observability and access controls. At the same time, Palantir's public tooling exposes parts of this ontology layer to external developers and agents through SDK and MCP-based interfaces, so it would be inaccurate to characterize the platform as entirely closed or inaccessible to third-party frameworks. The more precise distinction is architectural emphasis. Palantir is optimized for organizations that adopt Foundry as a primary operational and governance layer, with the strongest controls and capabilities expressed inside that environment. ContextOne, by contrast, is designed as a compositional and framework-agnostic control plane for heterogeneous agent estates: its ontology, memory, and governance layers are intended to span multiple agent frameworks, model providers, and data systems, including systems that remain outside any single vendor boundary.

Enterprise search and metadata management. A separate line of related work positions enterprise knowledge retrieval or metadata management as the foundation for AI-driven workflows. Glean constructs a comprehensive knowledge graph over enterprise content, people, and activity signals, combining vector, lexical, and graph-based retrieval with permission-aware retrieval-augmented generation to deliver contextualized answers across more than 100 SaaS connectors Glean (2024). In 2025, Glean introduced a multi-agent orchestration layer and an Enterprise Graph that merges organizational knowledge with personalized activity patterns, marking a transition from pure search toward agentic capability Glean (2025). Atlan takes a complementary approach as an active metadata platform: its Apache Iceberg-native Metadata Lakehouse provides an open, queryable metadata store with column-level lineage, automated governance, and—as of 2025—a Model Context Protocol (MCP) server that exposes the full metadata layer to external AI agents Atlan (2025). Both platforms contribute valuable infrastructure: Glean for permission-respecting content retrieval and Atlan for metadata enrichment and data governance. However, neither provides a general-purpose agent runtime. Glean's architecture remains oriented toward information discovery and has only recently begun supporting write operations in downstream systems; Atlan operates at the metadata layer rather than the operational data plane and does not perform cross-system entity reconciliation or federated query execution. ContextOne occupies a distinct architectural position as a runtime and control plane: it provides not only a federated data fabric and ontology but also a centralized Agent Harness with multi-layered memory (episodic, semantic,

and procedural), deterministic Named Query enforcement, and tunable autonomy controls that govern agent behavior across the full execution lifecycle.

Process intelligence and agentic analytics. Process mining and domain-specific analytics constitute a third category of related work. Celonis has advanced the field of object-centric process mining (OCPM), building a Process Intelligence Graph that models entities, events, and their relationships as a system-agnostic digital twin of operational workflows Celonis SE (2025). Van der Aalst has argued that such process intelligence is a prerequisite for effective enterprise AI, providing the organizational grounding that language models cannot infer from data alone van der Aalst (2025). With the AgentC initiative (2024) and a Model Context Protocol server (2025), Celonis has begun exposing process context to agents built on external platforms including Microsoft Copilot Studio, Amazon Bedrock, and CrewAI Celonis SE (2024). In the analytics domain, TextQL implements a three-layer architecture separating an ontology-based semantic layer (with deterministic metric definitions and join logic encoded in a proprietary Ontology Query Language), an agent layer with finite-state conversation management, and a sandboxed compute environment for SQL and Python execution TextQL (2025). TextQL's ontology addresses a genuine limitation of naïve text-to-SQL approaches: by encoding business definitions explicitly, the system ensures that identical questions yield identical queries regardless of the requesting agent Chen (2024). Both platforms, however, are domain-scoped—Celonis to process data within its ingestion boundary, TextQL to analytics over connected warehouses—and neither provides enterprise-wide agent governance, cross-domain memory, or formal verification of agent actions. ContextOne's architecture is domain-agnostic by design: its Named Queries enforce deterministic query semantics at the ontology level (analogous in principle to TextQL's metric definitions but generalized across all agent interactions and data modalities), while its Control Plane applies uniform governance across process, analytics, and operational workflows simultaneously.

Multi-agent governance, formal verification, and evaluation. The broader research and industry landscape underscores a growing consensus that governance and orchestration—not individual agent capability—constitute the primary barriers to enterprise adoption. Gartner projects that over 40% of agentic AI projects will be canceled by 2027 due to escalating costs, unclear value, or inadequate risk controls, while McKinsey's 2025 global survey found that only 6% of organizations qualify as AI high performers despite 88% reporting some form of AI adoption Gartner (2025); McKinsey & Company and QuantumBlack (2025). Recent academic work has begun to address specific facets of this challenge: Adimulam et al. survey multi-agent orchestration architectures with attention to policy enforcement, state management, and interoperability across MCP and A2A protocols Adimulam et al. (2026); Ehtesham et al. provide a comparative taxonomy of these emerging agent communication standards alongside ACP and ANP Ehtesham et al. (2025); and proposals for Governance-as-a-Service frameworks introduce modular enforcement proxies that filter agent actions against programmable rule specifications Pervez et al. (2025). A distinct and increasingly active thread of research explores the application of satisfiability modulo theories (SMT) solvers—specifically Z3—to AI safety and compliance, demonstrating their utility for verifying LLM-generated code, validating tool API safety, and enforcing regulatory constraints through neuro-symbolic integration Zhang et al. (2024); Untila (2026). ContextOne draws on this line of work directly: its Control Plane employs Z3-based constraint validation to enforce policy invariants over agent actions deterministically, moving beyond the purely empirical approaches (Palantir's AIP Evals, Celonis's process conformance checking) that characterize current industry practice. Its three-tiered evaluation framework (Instant Eval, LLM Judge, Agentic Grader) addresses the agent reliability gap documented by τ-bench, where even frontier models achieve sub-50% task success rates in enterprise-realistic interaction scenarios Yao et al. (2024).

7 Conclusion

The limitations of monolithic context management are now well understood. The initial wave of prompt-centric frameworks revealed the profound potential of agentic AI, but their architectural choices have stalled enterprise adoption. As organizations move from experimentation to production, the architectural choices made at this stage determine which systems scale and which stall. The evidence is clear: context management is an architectural concern, not a prompt-engineering one.

This paper introduced ContextOne, a system built from the ground up to address the enterprise challenges of scale, governance, and evaluation. We have detailed our two primary architectural contributions: the Agent

Harness, whose stateless execution model delivers accuracy and economic viability at massive scale, and our suite of Enterprise Controls, which provide the verifiable, deterministic trust required for mission-critical deployment. We have shown how these two innovations directly address the primary blockers to enterprise adoption. Furthermore, we have shared a new model for how to measure these systems, one that moves beyond academic benchmarks to account for the practical, economic realities of production AI.

Our primary lesson has been that treating context as a single, undifferentiated payload is a fundamental design flaw. By embracing a structured, multi-layered approach, we have shown that it is possible to build AI systems that are not only more powerful but also more reliable, auditable, and secure. The greater-than-3x improvement on the Tau Benchmark, achieved not by prompt-tweaking but by architectural rigor, validates this approach.

As organizations move from pilot projects to production-scale deployments, the architectural choices made today will determine which systems succeed and which fail. The next generation of enterprise AI will be built on platforms that treat context, control, and evaluation as first-class architectural concerns. ContextOne represents our contribution to this transformation, and the principles it embodies represent, we believe, a durable foundation for the future of enterprise AI.

Acknowledgements

We thank the AI One team and the broader enterprise AI community for their valuable feedback and contributions to this work.

References

A. Adimulam et al. The orchestration of multi-agent systems: Architectures, protocols, and enterprise adoption. *arXiv preprint arXiv:2601.13671*, 2026.

J. R. Aguilar. Crewai: A framework for orchestrating role-playing, autonomous ai agents. https://github.com/joaomdmoura/crewAI, 2023.

B. Atil, S. Aykent, A. Chittams, L. Fu, et al. Non-determinism of 'deterministic' llm settings. *arXiv preprint arXiv:2408.04667*, 2024.

Atlan. Data catalog architecture: Components, integrations, and design. Atlan Documentation, 2025. https://atlan.com/data-catalog-architecture/.

Celonis SE. Agentc: Making ai agents work for the enterprise with process intelligence. Press Release, October 23, 2024, 2024. https://www.celonis.com/news/press/celonis-agentc-making-ai-agents-work-for-the-enterprise-with-process-intelligence.

Celonis SE. Process intelligence graph. Celonis Platform Documentation, 2025. https://docs.celonis.com/en/process-intelligence-graph.html.

N. Chen. Why ontology for text-to-sql? TextQL Engineering Blog, 2024. https://nicholaschen.me/blogs/ontology-text-to-sql.

B. Dageville et al. The snowflake elastic data warehouse. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*, pages 215–226, 2016.

A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar. A survey of agent interoperability protocols: Mcp, acp, a2a, and anp. *arXiv preprint arXiv:2505.02279*, 2025.

Gartner. Top strategic technology trends for 2025: Agentic ai. Gartner Research, 2025. https://www.gartner.com/en/documents/5850847.

T. Ge, J. Hu, L. Wang, X. Wang, S.Q. Chen, and J. Sui. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Glean. Why great enterprise search and generative ai requires a comprehensive knowledge graph. Glean Engineering Blog, 2024. https://www.glean.com/blog/enterprise-ai-knowledge-graph.

Glean. Glean introduces third-generation ai assistant, new enterprise graph. BusinessWire, 2025. September 25, 2025. https://www.businesswire.com/news/home/20250925784461/en/.

R. Koohestani. Agentguard: Runtime verification of ai agents. *arXiv preprint arXiv:2509.23864*, 2025.

LangChain. Langgraph. https://langchain-ai.github.io/langgraph/, 2024.

P. Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

McKinsey & Company and QuantumBlack. The state of ai in 2025: Agents, innovation, and transformation. McKinsey Global Survey, November 2025, 2025. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai.

L. Mei et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

K. Milam and A. Gulli. Context engineering: Sessions, memory. Google Whitepaper, 2025. https://www.kaggle.com/whitepaper-context-engineering-sessions-and-memory.

Palantir Technologies. The ontology system. Palantir Architecture Center, 2024. https://www.palantir.com/docs/foundry/architecture-center/ontology-system.

Palantir Technologies. Aip architecture overview. Palantir Architecture Center, 2025. https://www.palantir.com/docs/foundry/architecture-center/aip-architecture.

S. Pervez et al. Governance-as-a-service: A multi-agent framework for ai system compliance and policy enforcement. *arXiv preprint arXiv:2508.18765*, 2025.

L. Qiu et al. Blueprint first, model second: A framework for deterministic llm workflow. *arXiv preprint arXiv:2508.02721*, 2025.

Snowflake. The open semantic interchange. https://www.snowflake.com/blog/open-semantic-interchange/, 2025.

TextQL. Agent architecture. TextQL Technical Documentation, 2025. https://docs.textql.com/core/how_it_works/architecture.

O. Untila. Emergent formal verification: How an autonomous ai ecosystem independently discovered smt-based safety across six domains. *arXiv preprint arXiv:2603.21149*, 2026.

W. M. P. van der Aalst. No ai without pi! object-centric process mining as the enabler for generative, predictive, and prescriptive artificial intelligence. In *Intelligent and Fuzzy Systems (INFUS 2025), Lecture Notes in Networks and Systems, vol. 1528*. Springer, 2025. DOI: 10.1007/978-3-031-97985-9_4.

S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Y. Zhang, Y. Cai, X. Zuo, X. Luan, K. Wang, Z. Hou, Y. Zhang, Z. Wei, M. Sun, Jun Sun, Jing Sun, and J. S. Dong. The fusion of large language models and formal methods for trustworthy ai agents: A roadmap. *arXiv preprint arXiv:2412.06512*, 2024.