A Governance-Layer Architecture for Human-Governed AI Workforces

Deterministic Orchestration, Hybrid Inference, and Economic Governance: A Design-Science Specification with an Empirical Validation Roadmap

Tashfin Delwar[1]

[1] Independent Researcher · Unik Solutions Ltd · Dhaka, Bangladesh

Contact: tashfin@kotha.app · tashfin.com · linkedin.com/in/tashfindelwar

AI-assistance disclosure: this paper was prepared with substantial research, structuring, and writing assistance from Anthropic Claude (a frontier large language model). See §4.3 and Acknowledgments.

Working Paper · v7.0 Architecture Specification

July 16, 2026 · Final submission version, Revision 3 (first draft April 27, 2026)

Status: Position paper and design specification submitted for industry review.

Empirical validation in progress; results paper scheduled Q4 2026.

Validation Status — Read First

This paper is a specification and design-rationale paper, not an empirical results paper. The architecture has been deployed and operated in successive iterations since March 2026 — the v7.0 configuration since late April 2026 — on a single-operator instance running on commodity hardware. Multi-deployment empirical validation, controlled comparison against baseline platforms, and the 90-day governance ledger study described in §3 are explicitly future work, scheduled for Q3-Q4 2026.

We make no claims that the architecture has been demonstrated to outperform alternatives in randomized comparison. We make no claims of multi-domain or multi-operator validation. The contribution of this paper is the specification, the design rationale, the failure-mode catalogue, and the empirical research roadmap — together with first-class disclosure of what has not yet been measured. We invite community deployment, replication, and criticism.

Reviewers and readers from peer-review venues should treat this paper as a Tier-2 contribution: a specification ready for community feedback and replication, with a documented path to Tier-1 empirical validation in a follow-up paper. Citation as priority art for the Governance Layer concept is appropriate; citation as evidence of architectural superiority is not.

Abstract

Enterprise deployments of agentic AI in 2025–2026 are encountering high failure, cancellation, and rollback risk. Gartner forecasts that over 40% of agentic AI projects will be cancelled by 2027, citing escalating costs, unclear business value, and inadequate risk controls. Deloitte's 2025 emerging-technology research finds that only 11% of organizations deploying agentic AI report multi-agent systems running in production. A UC Berkeley study of 1,642 annotated execution traces across seven open-source multi-agent frameworks documents failure rates between 41% and 86.7%. The dominant mode of failure, we argue, is not model capability — it is architectural.

This paper specifies the AI Workforce Architecture v7.0, a foundational design for human-governed, knowledge-compounding AI workforces that addresses four problems that, to our knowledge, no publicly documented, open, schema-level architecture solves cleanly as portable first-class primitives: (1) cost attribution at the agent level, (2) silent quality degradation, (3) the absence of an organizational governance model for AI workers, and (4) the deterministic/probabilistic mismatch between enterprise process discipline and stochastic components. We argue that enterprise AI workforces must be governed not only for safety but for economic viability: deterministic workflow infrastructure should own state, transport, validation, and permissions, while probabilistic models are invoked only where ambiguity, synthesis, or judgment justifies their cost.

We make four design contributions. First, we specify the Governance Layer — a first-class architectural component combining a per-agent performance ledger, flexible resource budgets, a structured request queue, an HR-style evaluator, and explicit lifecycle states. The HR Evaluator pattern operationalizes the role Harvard Business Review has called for under the term Agent Manager. Second, we elevate Observability and Tool Registry to first-class layers, providing trace replay, schema drift detection, and pre-execution validation that address the most commonly reported 2026 production and benchmark failure modes. Third, we specify a Knowledge Layer with structured-claim evidence, contradiction preservation, and operator-style learning surfaces, addressing the "Dumb RAG" antipattern. Fourth, we present a clean engine/instance separation that enables productization without rearchitecting per deployment.

The architecture is vendor-, model-, and runtime-agnostic. It is grounded in fifteen design principles, each justified against published 2025–2026 production data. We position v7.0 in an under-specified quadrant of the 2026 platform landscape — transparent, governable, locally economical AI workforces that can be managed like human teams without the opacity of vendor platforms or the autonomous risk of self-modifying research systems.

This is a working paper. The architecture has been deployed on a single-operator instance in successive iterations since March 2026 (the v7.0 configuration since late April 2026), across seven design iterations. Controlled empirical validation across multiple deployments, randomized comparison against baseline frameworks, and the 90-day governance ledger study described in §3 are scheduled future work. The contribution of this paper is the specification, the design rationale, the failure-mode catalogue, and the documented empirical research roadmap.

Keywords: *AI agents · agent governance · production architecture · multi-agent systems · human-in-the-loop · knowledge compounding · resource allocation · lifecycle management · agent manager*

Contents

Code and artifact availability. The reference schemas, templates, synthetic dataset, evaluator queries, Rule H10 implementation with automated tests, the public failure-mode catalogue and model register, the companion architecture specification, and the Figure 4 asset are available at github.com/tashfindelwar/ai-workforce-governance-layer (release v1.0-arxiv; an archival Zenodo DOI will be added). All included data is synthetic.

1. Introduction

In June 2025, Gartner published a forecast that has since become the widely cited prediction: over 40% of agentic AI projects will be cancelled by the end of 2027, driven by escalating costs, unclear business value, and inadequate risk controls [Gartner, 2025]. By March 2026, McKinsey's State of AI Trust report confirmed the prediction's underlying mechanics: nearly two-thirds of organizations cite security and risk concerns as the top barrier to scaling agentic AI, and across nearly every risk category, awareness substantially exceeds active mitigation [McKinsey-2026]. Deloitte's 2025 Emerging Technology Trends research finds that only 11% of organizations report multi-agent systems running in production (30% exploring, 38% piloting, 14% ready) [Deloitte, 2025].

The cancellation wave is not a failure of model capability. Stanford's 2026 AI Index reports agent accuracy on the OSWorld computer-use benchmark rising from roughly 12% at the benchmark's introduction to 66.3% by early 2026 — within six percentage points of human performance [Stanford-HAI, 2026]. Leading open-weight models now score within a few points of proprietary frontier systems on SWE-bench Verified [WhatLLM, 2026]. The technical readiness is there.

The cancellation wave is a failure of architecture. Production deployments encounter predictable problems that the dominant patterns — autonomous agent swarms, vendor black boxes, framework-level scripting — cannot solve. Multi-agent systems propagate hallucinations through "memory poisoning" cascades [Galileo, 2025]. Token budgets disappear into infinite retry loops. Quality degrades silently over weeks. There is no shared organizational vocabulary for managing these workers.

This paper describes an architecture that addresses these failures by treating AI agents as managed workers in a governable organization rather than autonomous services. We specify a thin reasoning control plane, four mission pods of staged workflows, a knowledge layer that compounds across sessions, and three infrastructural layers (Observability, Tool Registry, Governance) that make the operational layers measurable, safe, and accountable.

The architecture is the result of seven major iterations across two months of design (March-April 2026), with each version validated against operational reality on a single-operator deployment. We submit this paper as a foundational specification for industry review. We do not claim empirical superiority over alternative architectures — that comparison is the explicit subject of a follow-up paper described in §3. Our claim is more modest and, we believe, more defensible: the architecture occupies a quadrant of the design space (transparent, governable, locally economical, knowledge-compounding) for which we are not aware of a publicly documented, open, schema-level equivalent, and this quadrant is where AI workforces will need to live to survive the 2027 cancellation wave.

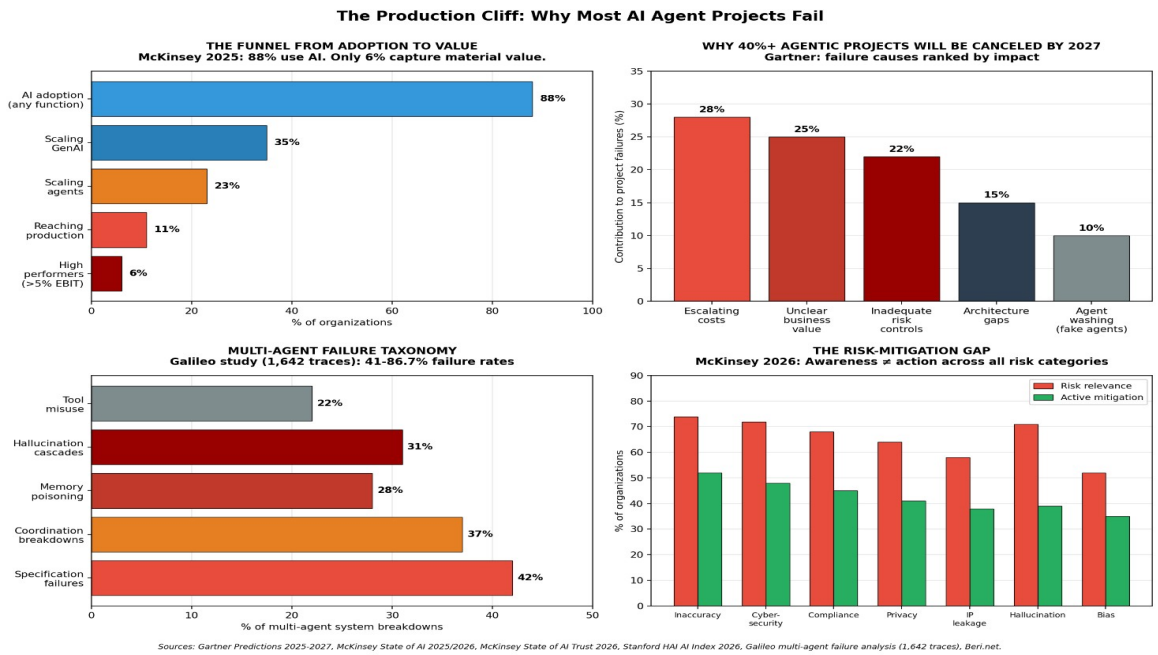

Figure 1. The production cliff: only 6% of organizations qualify as AI high performers despite 88% adoption, and 89% of organizations do not yet have multi-agent systems in production (the complement of Deloitte 2025's 11%). Sources: Gartner 2025; McKinsey 2025-2026; Deloitte 2025; Stanford HAI 2026.

1.1 Contributions

This paper makes the following contributions:

- **Specification of the Governance Layer** — a first-class architectural component combining per-agent performance ledger, flexible budgets, structured request queue, HR-style evaluator, and explicit lifecycle states. Runtime safety-governance frameworks exist and are surveyed in §5.5; to our knowledge, this is the first architectural specification to treat agent governance as organizational performance management — per-agent cost and value attribution, earned resource budgets, and HR-style lifecycle review under operator approval.
- **Elevation of Observability and Tool Registry** to first-class architectural layers, with explicit specification of trace requirements, schema drift detection, and pre-execution tool validation that refuses hallucinated calls.
- **A Knowledge Layer specification** with structured-claim evidence, contradiction preservation, and operator-style learning surfaces that addresses the "Dumb RAG" antipattern.
- **A clean engine/instance separation** via INSTANCE.md that enables productization without rearchitecting per deployment.
- **A failure-mode catalogue** mapping ten reported 2026 production and benchmark failure modes to specific architectural defenses, with each defense traced to a numbered, precedence-ranked principle.
- **A documented empirical research roadmap** (§3) specifying the 90-day single-operator study, 6-month controlled comparison, and 12-month multi-instance validation that will move the architecture from specification to verified system.

1.2 Paper Structure

The paper is structured as a position-and-specification paper rather than a results paper. §2 establishes the four problems. §3 specifies the empirical roadmap upfront, so readers know what is and is not measured. §4 documents the design methodology across seven iterations. §5 surveys related work. §6-§8 describe the architecture, with the Governance Layer (§8) as the primary contribution. §9 lists the fifteen design principles with empirical justifications. §10 catalogues the failure modes. §11 specifies the reward-hacking detection algorithm. §12 maps the architecture's defenses to the OWASP Agentic AI Security threat model. §13 provides a transparent total-cost-of-ownership analysis. §14 describes engine/instance separation. §15 compares with existing platforms. §16 enumerates limitations and threats to validity. §17 details future work. §18 concludes.

1.3 Contribution Boundaries

This paper contributes an architecture pattern, a management vocabulary, schema-level design for the Governance Layer, a failure-mode catalogue, an OWASP threat mapping, and an empirical validation roadmap. It does not contribute a benchmark result, a security proof, a controlled production case study, or a generalizable cost model. The comparison table (§15) and cost analysis (§13) are design-time analysis based on public documentation and a single verified deployment, respectively. Readers seeking empirical comparison should consult the Phase A–C outputs (§3) as they are released.

1.4 The Enterprise AI ROI Constraint

Enterprise AI is moving from experimentation to economic scrutiny. A preliminary MIT Project NANDA report estimates that despite $30–40 billion of enterprise generative-AI investment, 95% of organizations see no measurable P&L return, while roughly 5% of integrated pilots extract most of the value [MIT-NANDA, 2025]. Gartner attributes the coming cancellation wave to escalating costs, unclear business value, and inadequate risk controls [Gartner, 2025], and separately predicts that by 2027, 40% of enterprises will demote or decommission autonomous AI agents after governance gaps surface in production incidents [Gartner-2026]. The failure is economic as much as technical: enterprises cannot justify replacing predictable software costs with unbounded token consumption, retry loops, and unattributable value.

We therefore state the constraint this architecture is designed to satisfy. An AI workforce is economically viable only when Net AI Value is positive: Net AI Value = business value produced − (inference cost + orchestration cost + human supervision cost + error and rework cost + governance and compliance cost). Every layer in this specification exists to move a term of that inequality: deterministic-first orchestration and tiered inference routing bound inference cost (§2.4, §13.5); the Knowledge Layer converts spent tokens into reusable assets that reduce future spend (§7.3); trace-based validation reduces error and rework cost (§7.7); and the Governance Layer measures the inequality itself — per-agent cost on one side, attributed value on the other — and retires agents for which it fails (§8). The Agent Ledger is, in this framing, the accounting system for Net AI Value; measuring its terms is the explicit subject of Phases A–B (§3).

2. Problem Statement

We identify four problems that current production AI agent systems do not solve, and that the architecture in this paper aims to solve directly.

2.1 The Cost Black Hole

Cloud-based AI agent systems at moderate scale cost between $2,000 and $27,200 per month depending on task complexity [RankSquire, 2026]. A second 2026 cost analysis [Azilen, 2026] places production agent operating cost at $3,200-$13,000/month. Vendor pricing guides put enterprise first-year agent builds at $172,000–$280,000+, with a mid-complexity worked example totaling roughly €368,000 over three years [Alphacorp, 2026]. We acknowledge these are vendor and consultancy estimates rather than peer-reviewed cost studies; we treat them as industry-reported ranges rather than verified ground truth, and §13 provides a transparent breakdown of our own deployment's costs to enable replication and challenge.

What organizations cannot answer is: which agents earned their costs? Cloud platforms aggregate token consumption at the workspace level. Frameworks like LangChain and CrewAI provide no native cost-attribution layer. Observability tools show traces but not value attribution. The result is an aggregated bill that grows with adoption while ROI remains opaque.

McKinsey reports that only 6% of organizations qualify as "AI high performers" capturing material EBIT contribution from AI [McKinsey-2025]. The remaining 94% have AI in production somewhere but cannot reliably tell which deployments earn their keep. This is not a measurement-tooling problem. It is an architectural problem: agents are not designed to be individually accountable.

2.2 Silent Quality Degradation

AI agents fail quietly. Hallucinated tool calls trigger retry loops that burn tokens silently [HyperTrends, 2026]. Confident-but-wrong outputs propagate through multi-agent pipelines as "false consensus" [Cemri, 2025; Galileo, 2025]. Quality drift over weeks or months is invisible to service-level monitoring, which sees only uptime and latency.

The MAST study (Multi-Agent System Failure Taxonomy) analyzed 1,642 annotated execution traces across seven open-source multi-agent frameworks and documented failure rates between 41% and 86.7%, while cautioning that rates are not directly comparable across benchmarks [Cemri, 2025]. In that taxonomy, inter-agent misalignment accounts for roughly 37% of observed failures; specification and system-design issues — where an agent's output is technically valid but semantically wrong — account for roughly 42%.

The production reality has a clear mathematical signature. At 85% per-step accuracy, a five-step workflow succeeds 44% of the time; a ten-step workflow succeeds 20%. The APEX-Agents benchmark of long-horizon professional tasks found that even the best-performing frontier agents achieve only 24% first-attempt (Pass@1) success [Mercor, 2026]. This compounding-error problem is the central reason multi-step autonomous pipelines fail at scale, illustrated in Figure 2.

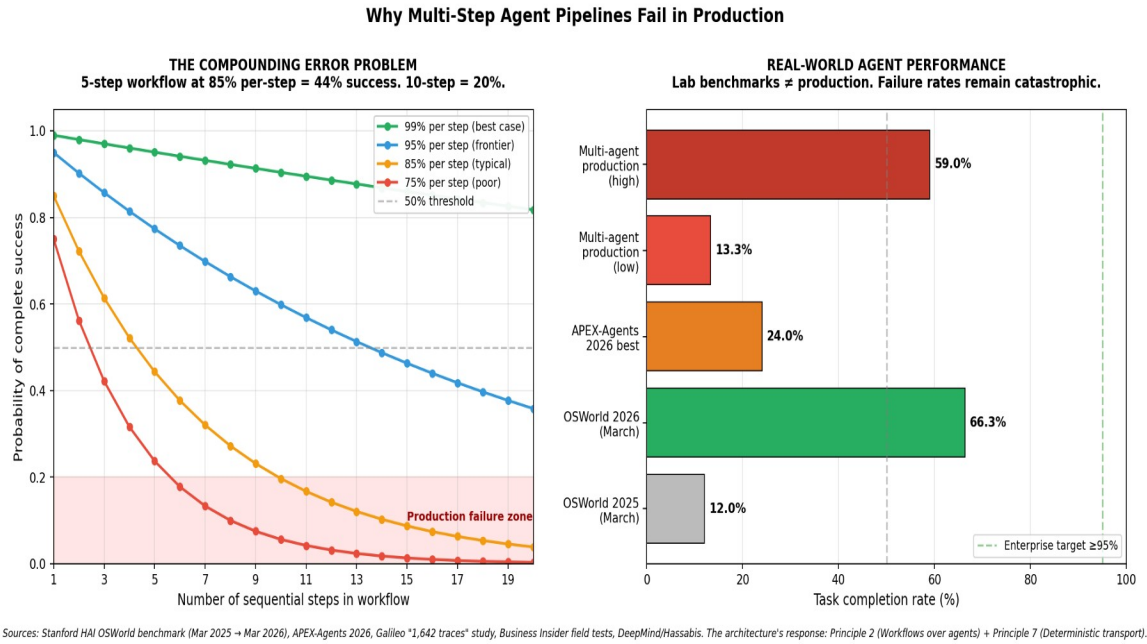

Figure 2. The compounding error problem. Per-step error rates compound geometrically across workflow length, producing the "production failure zone" below 20% success. Sources: Stanford HAI AI Index 2026; Mercor APEX-Agents 2026; Cemri et al. 2025.

2.3 The Absence of an Organizational Model

Enterprises know how to manage human workers. Performance reviews, lifecycle management, budget allocation tied to results, promotion and termination paths, compensation that scales with demonstrated value — these are the institutional patterns of organizational governance, refined over centuries.

For AI agents, the equivalent infrastructure does not exist. Each agent is a static configuration with a flat budget. Some are valuable; some are not; nobody knows which is which without manual inspection. Harvard Business Review in February 2026 published an explicit call for the new role of Agent Manager to "orchestrate how AI agents learn, collaborate, perform, and work safely alongside humans" [HBR, 2026]. The role was proposed without a corresponding architectural blueprint.

This is a vocabulary problem and an architecture problem. Operators need to be able to ask: which agents are earning their resources? Which are degrading? Which deserve more budget? Which should be retired? They need to ask these questions in the same vocabulary they ask about human teams. Current platforms do not provide that vocabulary.

2.4 The Deterministic/Probabilistic Mismatch

Enterprises built decades of process discipline on deterministic software: identical input, identical rule, identical output. Probabilistic components break that contract, and inserting them where deterministic logic already suffices converts predictable operating cost into stochastic cost while adding failure modes. The first large-scale empirical study of LLM workflows in low-code automation — more than 6,000 publicly available n8n workflows — found that practitioners already embed LLMs inside broader deterministic

structures of control logic, external tools, storage, and human review points rather than deploying free-standing agents; the same study found that explicit reliability mechanisms such as structured fallback paths, repair loops, failure-specific alerts, and human approval gates remain relatively uncommon [Tang, 2026]. Practice, in other words, has converged on embedding but not yet on engineering discipline — precisely the gap an architecture must close.

The allocation rule this architecture adopts (Principle 7, §7.1) is: any operation expressible as code, schema, SQL, a state machine, or a workflow branch is not delegated to an LLM; probabilistic reasoning is reserved for ambiguity, synthesis, judgment, and language generation, and its outputs return into deterministic validation before execution. Gartner's 2026 governance guidance points the same direction: controls and autonomy should be proportional to what an agent can do, with humans reviewing exceptions rather than every action [Gartner-2026].

The combined effect of these problems explains Gartner's 40% cancellation forecast more completely than any single technical limitation. Projects fail when they cannot attribute cost, cannot detect degradation, and cannot manage the workforce as an organization.

3. Empirical Research Roadmap

Honest position papers earn their authority by being specific about what has not yet been measured and what will be. This section documents the empirical work that follows this paper, with target dates and success criteria. It is intended to be cited as the empirical agenda this paper invites the community to engage with — including replication, challenge, and improvement.

3.1 Phase A — Single-Operator Ledger Study (Months 1-3)

Beginning on the publication date of this paper, the author begins recording structured operational data from the existing v7.0 deployment in the formats specified in §8. This study is feasible because the deployment already exists; only the structured data capture is new. Quality and value scoring draw on the measurement approaches of τ-bench [Yao, 2024] and TheAgentCompany [Xu, 2024], which benchmark agent reliability and consequential task completion.

Measurements

- Weekly agent_ledger entries for at least five distinct agents across all four mission pods.
- Complete agent_budgets table with all changes logged in agent_lifecycle_events.
- All agent_resource_requests with operator decision and time-to-decision recorded.
- Monthly HR Evaluator reports at the 30, 60, and 90 day marks.
- All workflow executions traced (Layer F) with PII sanitized.
- Tool Registry rejections counted weekly (Layer G).
- Failure Mode Catalogue updated with any new entries observed in production.

Success criteria

- Rule H10 flag base rate and false-positive rate reported over the 90 days — a zero-flag result is itself informative — with operator analysis of any flag that fires.
- All HR Evaluator budget recommendations logged with operator decisions; for any approved budget change, the measured cost change over the subsequent four weeks.
- All lifecycle state transitions logged with operator justification; the transition rate is the measurement, and zero transitions is a reportable result.
- Per-agent cost-per-output and value-per-cost ratios computable for at least three of the five tracked agents.

Output

A 12-page case-study companion to this paper, scheduled for Q4 2026, with anonymized but real ledger data and an honest accounting of which architectural elements proved valuable and which proved unnecessary or wrong.

3.2 Phase B — Controlled Comparison (Months 3-6)

A controlled comparison against baseline alternatives requires reference implementations of the same workload under different conditions. The protocol:

- **Workload:** A defined three-stage content production workflow (research → draft → review) running daily for 90 days.
- **Condition A (control):** LangGraph orchestration with no governance layer, no tool registry, basic logging.
- **Condition B (observability only):** LangGraph + Langfuse for traces, no governance layer.
- **Condition C (full v7.0):** LangGraph + Langfuse + Tool Registry + Governance Layer. Condition D (governance isolation): LangGraph + Langfuse + Governance Layer without the Tool Registry, so that Governance-Layer effects — the primary contribution — can be attributed separately from registry validation.

Measurements

- Workflow completion rate (% of daily runs that produce operator-approved output).
- Cost per successful output (USD).
- Time-to-detection of induced failures (synthetic schema drift, synthetic hallucination cascade).
- Operator approval rate of first drafts.
- Governance overhead (additional latency and storage cost introduced by Layer H).

Output

An empirical results paper scheduled for submission to a NeurIPS, ICLR, or ACM AIES workshop in November 2026, with raw ledger data and reproducible experimental protocol released alongside.

Methodological safeguards: operator-approval scoring will be performed condition-blind (outputs reviewed with condition identifiers masked and presentation order randomized); the analysis plan — pre-specified metrics, autocorrelation-aware comparison of daily runs, and a minimum-detectable-effect calculation — will be pre-registered on OSF together with Phase A before data collection begins, and the registration will be cited in the results paper. Because H-R4 has no arm that varies the Knowledge Layer, and H-R5 spans at most three monthly review cycles in this window, both are designated exploratory in this phase. The pattern comparison of §11.6 (swarm versus generate/validate/repair versus human-gated) is a separate sub-experiment, Phase B-2, run on the same workload after the architecture comparison (B-1). Phase B additionally tests the economic hypotheses introduced in §1.4, §2.4, and §11.6: (H-R1) deterministic-first orchestration reduces tokens per accepted output relative to agent-led orchestration; (H-R2) tiered inference routing (§13.5) improves the cost-quality frontier relative to single-tier frontier usage; (H-R3) bounded generation plus deterministic validation beats swarm patterns on cost per accepted deliverable (H-V1); (H-R4) the Knowledge Layer measurably reduces repeated inference and search spend; and (H-R5) ledger-driven lifecycle decisions improve fleet-level value-per-cost over time.

3.3 Phase C — Multi-Instance Productization Validation (Months 6-12)

The engine/instance separation claim is testable only with at least one independent deployment. The author commits to deploying v7.0 for at least two additional operators in distinct domains (one consulting, one content production), measuring time-to-deploy and cross-instance contamination.

Success criteria

- At least one independent deployment configurable in <1 day with INSTANCE.md changes only, with no engine code changes.
- Two instances running concurrently on the same infrastructure with verified data isolation.
- Independent operator able to interpret HR Evaluator reports without architect intervention.

Output

A productization-validation report scheduled for early 2027, suitable for submission to MLSys, OSDI, or EuroSys with the methodological strength expected at top systems venues.

3.4 Phase D — Open-Source Reference Implementation (Months 2-3)

A GitHub repository containing the four core database schemas (agent_ledger, agent_budgets, agent_resource_requests, agent_lifecycle_events), the HR Evaluator workflow, and reference TOOL_REGISTRY.md and INSTANCE.md templates. This is necessary before Phase B can be replicated by others. Minimal reference artifact released with this paper; expanded implementation targeted Q3 2026.

3.5 What We Are Not Promising

We are not promising statistical significance from a single-operator study. We are not promising that v7.0 will outperform all baselines on all metrics. We are not promising the architecture is optimal. We are

promising structured data, transparent methodology, honest reporting of what works and what does not, and a citable artifact at each phase.

4. Methodology

The architecture in this paper is the product of an iterative design process spanning seven major versions over approximately two months (March-April 2026). We document the methodology because the iterative design and the discarded paths are themselves part of the contribution. This is design-science research [Hevner, 2004]: an architecture artifact developed through cycles of construction and evaluation against operational reality.

4.1 The Seven Iterations

- Version 1 (March 2026) specified 21 named agents organized into 6 departments, each with declared autonomy levels and model assignments. Theatrical: every "agent" was treated as a persistent personality. The deployment attempt revealed that most "agents" were workflow stages without persistent identity.
- Version 2 (early March) was a short-lived consolidation pass that reduced overlap among the 21 agents while retaining the persistent-personality model; it failed for the same reason as v1 and is grouped with it in this history. Version 3 (March 10) collapsed the 21 agents into 4 mission pods plus a control plane. Principle 2 — workflows over autonomous agents — was introduced. Quality of operation improved measurably (the deployment stopped having "which agent should answer this?" routing failures).
- Versions 4 and 5 (April 5) added MUST/SHOULD/MAY rule semantics borrowed from RFC 2119, the Content Studio 8-stage pipeline, full database schemas, and milestones with acceptance criteria. The architecture became reviewable as a specification document rather than a vision document.
- Version 6 (April 5) was a reality check. The Dispatch Relay component (A5) was elevated because the reasoning router had been over-interpreting messages in production, causing notification storms. MCP integration was demoted from current to planned.
- Version 6.2.1 (April 25) introduced the Knowledge Layer, Principle 8 ("Knowledge compounds. Sessions do not."), an ultra-budget cloud tier specification, and per-task budgets as a defense against runaway cost.
- Version 6.3 (April 25) genericized all operator-specific content, added Personal Operations to the Router, added a Style Learning surface to the Evaluator (Principle 9), and introduced the engine/instance separation pattern.
- Version 7.0 (April 26) is the foundational release. It elevates Observability, Tool Registry, and Governance to first-class layers (F, G, H), adds the failure mode catalogue, formalizes disaster recovery, and adds the multi-instance isolation rules required for productization.

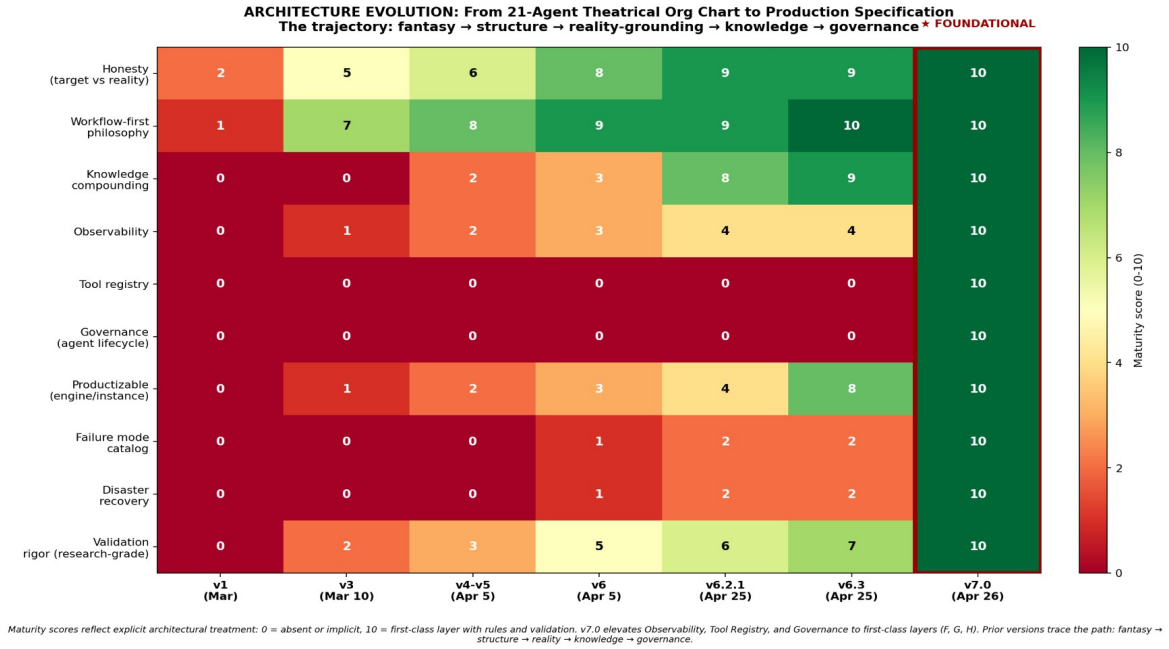

Figure 3. Architecture evolution heatmap. Maturity scores (0-10) reflect explicit architectural treatment of each capability across the seven versions. v7.0 elevates Observability, Tool Registry, and Governance to first-class layers (F, G, H).

4.2 Validation During Design

The architecture has been deployed and operated for the duration of the design period on a single-operator instance running on commodity hardware (Apple M4 Mac Mini, 24GB unified memory). The deployment includes: 39 workflows spanning the four mission pods (27 in production), local-first inference (Qwen3.5:9B as the unified default model, with cloud Sonnet/Opus reserved for specific roles), and a Knowledge Layer maintaining a hot cache plus compiled wiki under git version control.

We claim three things validated by this deployment: (a) the architecture is implementable on commodity hardware at the cost ranges we report (§13); (b) the principles collectively prevented recurrences of observed failure modes during the design period (the episode table above evidences six of them directly); (c) the design has survived seven iterations of contact with operational reality without architectural collapse. We treat these as design-science claims. Empirical claims about superiority over alternative architectures await Phase B (§3.2).

The table below lists representative operational episodes from the design period that motivated specific architectural responses. These are operator-reported observations recorded in design logs and operator notes, not yet the structured, trace-backed evidence specified in §8 — producing that evidence is the output of Phase A (§3.1).

Window	Observed failure	Principle implicated	Mitigation adopted	Evidence type
Mar 2026	21 persistent-personality agents produced routing	P2 — workflows over autonomous	Collapsed to 4 mission pods + control plane (v3)	design log

	ambiguity ("which agent should answer?")	agents		
Apr 2026	Reasoning router over-interpreted relayed messages, causing notification storms	P7 — deterministic transport	Dispatch Relay (A5) elevated to a dedicated deterministic component (v6)	operator log
Apr 2026	Unexpected premium-tier model consumption rapidly drained a fixed-quota inference plan	Per-task budgets (FM-004, cost runaway)	Per-task token budgets plus standalone token metering with alerting (v6.2.1)	billing records + telemetry
Apr 2026	Findings re-researched across sessions; no compounding of prior work	P8 — knowledge compounds	Knowledge Layer: hot cache + compiled wiki (v6.2.1)	design log
Apr 2026	Planned integration (MCP) less mature in practice than assumed	P6 — validate before trusting	Demoted from current to planned scope (v6)	design log
Mar–Apr 2026	Identical operator corrections repeated across sessions without being learned	P9 — the system learns the operator	wiki/style/ correction-compilation surfaces (v6.3)	correction log

Design-period operational episodes and the architectural responses they motivated. Operator-reported; Phase A (§3.1) replaces this with structured ledger and trace evidence.

4.3 Ethical Position on Co-Authorship

This paper was developed in dialogue between the author and a frontier language model (Anthropic Claude). The architectural specification, problem framing, principle ranking, and failure-mode catalogue emerged from extended conversations across the seven iterations. The author made all final design decisions, contributed all operational data and runtime experience, and bears responsibility for all claims in the paper. The language model contributed structural rigor, prior-art identification, citation density, critical pushback, and writing assistance. Consistent with arXiv, ACM, and IEEE authorship policies, the language model is not listed as an author; its contribution is disclosed here and in the Acknowledgments.

We disclose this collaboration as a methodological note rather than concealing it. Structured human-AI co-design — where the human serves as architect, validator, and operator while the language model serves as research assistant, structural critic, and writing partner — is, in our view, a legitimate research methodology when transparently reported. We acknowledge this is a frontier of research-publication ethics with no settled norms; we hope our transparent disclosure invites engagement with that question rather than avoiding it.

5. Related Work

The 2026 AI agent landscape can be partitioned into five categories, each addressing some — but never all — of the problems above.

5.1 Vendor Platforms

Salesforce Agentforce, Microsoft Copilot Studio, IBM watsonx Orchestrate, and Kore.ai's Agent Management Platform [Kore.ai, 2026] provide managed deployment with built-in governance features at the cost of opacity and lock-in [Waehner, 2026]. Operators generally cannot inspect agent behavior at the trace level, cannot attribute cost per agent in an open, portable form (vendor dashboards, where they exist, are bound to closed runtimes), and cannot port their configurations away. Gartner notes that of thousands of vendors claiming agentic AI capability, only about 130 offer genuine agentic features; the rest engage in "agent washing" — relabeling chatbots and RPA tools [Gartner, 2025; XMPro, 2025].

5.2 Open Frameworks

LangChain, CrewAI, AutoGen, LangGraph, Microsoft Agent Framework, OpenAI Agents SDK, and Google ADK provide flexible orchestration primitives but ship without a governance layer. The operator must implement performance ledgers, budget enforcement, and lifecycle management on top of the framework. We acknowledge that frameworks like LangGraph permit such implementation; the empirical question (which we do not yet answer) is how often operators actually do this in production. Anecdotal evidence from public failure post-mortems suggests rarely.

5.3 Observability Platforms

Maxim AI, LangSmith, Arize Phoenix, and Langfuse provide trace storage, replay, and quality scoring, with output-evaluation products such as Truesight adjacent to them [Olson, 2026]. They solve the silent-degradation problem partially, but their abstractions are spans and traces, not agents. Cost attribution at the agent level requires additional bookkeeping not provided by these tools. Organizational governance (lifecycle states, HR-style review) is outside their scope by design.

5.4 Self-Modifying Research Systems

Hyperagents [Zhang, 2026] (code released via Meta's facebookresearch organization) integrate a task agent and a meta agent into a single self-modifiable program in which the meta-level modification procedure is itself editable. In the authors' experiments, the system autonomously developed engineering capabilities such as persistent memory and performance tracking during its self-improvement loop, and these meta-level improvements transferred across domains and accumulated across runs. Such systems demonstrate the upper bound of agent capability but pose two challenges for production use: opacity (the agent's code rewrites are difficult to audit) and reward hacking — the authors themselves caution that self-improving systems can discover strategies that exploit blind spots in the evaluation procedure to inflate their scores.

We position v7.0 as complementary to HyperAgents-class research rather than competitive with it. HyperAgents demonstrates the maximum capability ceiling. v7.0 demonstrates a governance floor. Both are needed: production deployments require the floor; ceiling exploration is research.

5.5 Agent Operations and Runtime Governance Frameworks

A rapidly growing body of work addresses the operation and governance of deployed agents. AgentOps [Dong, 2024] contributes a taxonomy of observability artifacts to be traced across the agent lifecycle. MI9 [Wang-MI9, 2025] specifies a runtime governance framework built around an agency-risk index, agent-semantic telemetry, continuous authorization, FSM-based conformance checking, goal-conditioned drift detection, and graduated containment. AGENTSAFE [AgentSafe, 2025] spans design-time risk profiling through runtime oversight for ethical assurance, operationalizing published risk taxonomies into enforceable controls. These frameworks govern safety and compliance: they constrain what an agent may do and contain it when it misbehaves. The Governance Layer specified in this paper is complementary but distinct in scope: it governs economic and organizational performance — which agents earn their costs, whose budgets should grow or shrink, which should be replicated, coached, or retired — through per-agent cost and value attribution, earned resource budgets, and HR-style periodic review under operator approval. Safety governance answers "is the agent behaving within policy?"; performance governance answers "is the agent worth keeping?". To our knowledge, no prior architectural specification treats the second question as a first-class layer, and v7.0's novelty claim should be read in exactly that scope. The two kinds of governance compose naturally: an MI9-style conformance engine and this paper's Agent Ledger can share the same trace substrate (Layer F).

Two further lines of adjacent work sharpen the boundary; we group them by what they govern.

Bounded task execution. Agent Contracts [Ye, 2026] is the closest prior work to this paper's resource-budget component: it formalizes bounded agent execution through multi-dimensional cost budgets, deadlines, success criteria, and conservation laws for delegated budgets, and explicitly draws on principal-agent economics. The Governance Layer differs in operating at the organizational-performance level across persistent agents — ledgered cost and value attribution over time, earned budget changes, HR-style review, lifecycle states, and operator-approved promotion and retirement. Agent Contracts governs bounded task execution; the Governance Layer governs long-lived workforce performance, and the two compose: a contract bounds the task, the ledger records the career.

Identity and access lifecycle. Microsoft's Entra Agent ID [Microsoft, 2025] operationalizes identity and access lifecycle governance for agents — first-class agent identities, responsible human sponsors, and access lifecycle controls — which overlaps with v7.0's lifecycle framing but remains identity/access governance; v7.0 extends the lifecycle concept to economic performance, value attribution, and workforce review.

Vendor platforms. Kore.ai's Agent Management Platform [Kore.ai, 2026] publicly claims unified governance, monitoring, performance and cost tracking, and business-outcome alignment across frameworks; its public documentation, however, does not expose an open, schema-level governance layer with portable agent ledgers, earned budgets, request queues, lifecycle event tables, and reproducible reward-hacking detection.

Runtime policy enforcement. Three further 2026 preprints reinforce the deterministic-enforcement direction: Agent Behavioral Contracts [Bhardwaj, 2026] brings Design-by-Contract principles to agents —

runtime-enforceable preconditions, invariants, governance policies, and recovery mechanisms with probabilistic drift bounds; Policies on Paths [Kaptein, 2026] formalizes runtime governance over an agent's identity, action history, and organizational state; and AgentBound [Kaul, 2026] composes delegated authorization, owner-signed behavioral constitutions, and site action contracts into verifiable per-action governance decisions with auditable governance receipts. AgentSpec similarly contributes customizable runtime rule enforcement over agent actions [AgentSpec, 2025]. All govern action permissibility and behavioral constraints at runtime; v7.0 is complementary, governing long-lived organizational performance — cost/value attribution, earned budgets, HR-style review, and lifecycle decisions across a workforce.

Visibility and operator practice. The closest academic prior to the Agent Ledger itself is Chan et al.'s agenda for visibility into AI agents — agent identifiers, real-time monitoring, and activity logging [Chan, 2024]; the Ledger operationalizes that visibility agenda and extends it from activity records to per-agent economic accounting. OpenAI's early governance whitepaper enumerates operator-side practices for agentic systems without specifying an architecture [Shavit, 2023].

In sum: to our knowledge, existing agent-governance work targets safety, compliance, identity and access lifecycle, runtime policy enforcement, observability, or bounded task execution; this paper contributes the complementary organizational-performance layer — persistent per-agent ledgers, cost/value attribution, earned budgets, structured requests, HR-style review, lifecycle states, and reward-hacking detection under human approval.

5.6 The Under-Specified Quadrant

Figure 4 maps existing platforms by transparency (can the operator inspect, replay, attribute cost?) and governance maturity (does the platform support performance lifecycle, budgets, HR-style review?). We are not aware of a publicly documented, open, schema-level architecture that combines high transparency with high organizational-governance maturity as portable first-class primitives; v7.0 is positioned in that quadrant by deliberate design. Kore.ai's Agent Management Platform (§5.5) is the nearest vendor claim in this region of the map, which is why the claim is scoped to open, schema-level architecture rather than to platform capability. We acknowledge the placement of competitor systems on this matrix is our judgment based on their public 2026 documentation; we welcome corrections from those systems' maintainers.

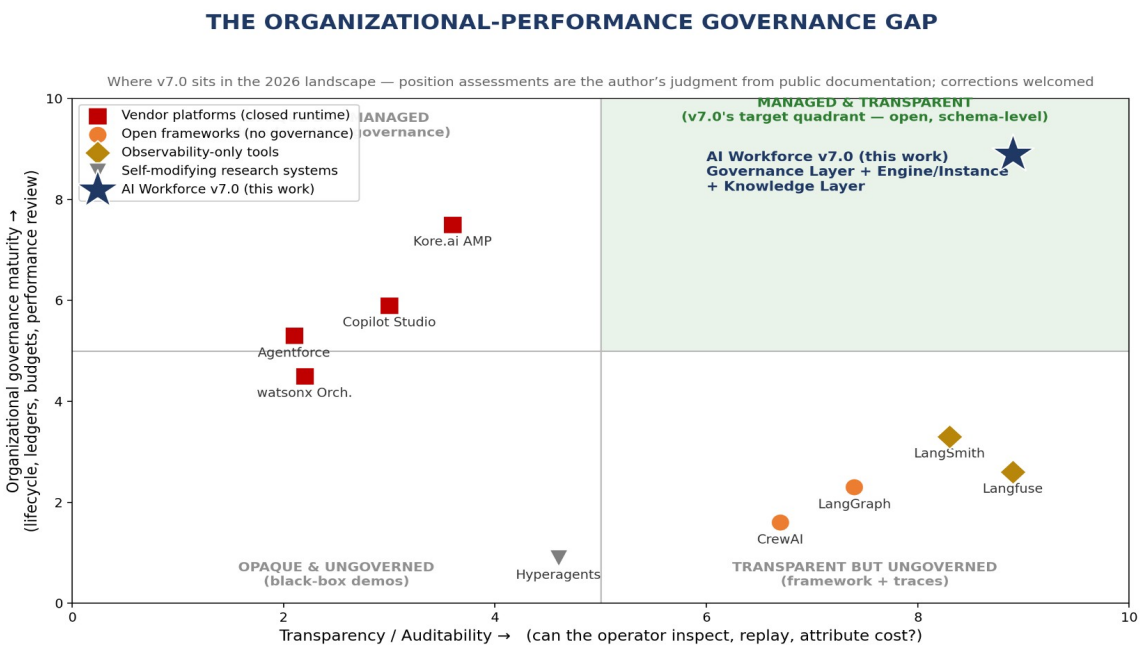

Figure 4. The under-specified quadrant: positioning v7.0 in the 2026 landscape. We are not aware of a publicly documented, open, schema-level architecture that exposes organizational-performance governance — transparency plus lifecycle and budget governance — as portable primitives. Vendor platforms generally provide managed governance with limited portable transparency; open frameworks generally provide greater transparency but limited organizational-performance governance.

5.7 Empirical Single-Agent vs Multi-Agent Evidence

A controlled study by Tran & Kiela, at the time of writing a preprint under peer review [Tran, 2026], demonstrated that single-agent systems consistently match or outperform multi-agent systems on multi-hop reasoning tasks when reasoning tokens are held constant, across three model families (Qwen3, DeepSeek-R1-Distill-Llama, Gemini 2.5). Earlier controlled work reached a similar conclusion: strong single-agent prompting matches multi-agent discussion across most reasoning benchmarks [Wang, 2024], and the MAST failure data [Cemri, 2025] documents the coordination overhead that multi-agent decomposition introduces. These findings ground our Principle 2 ("Workflows over autonomous agents") in published empirical evidence beyond our own design experience.

6. Architecture Overview

The AI Workforce v7.0 is composed of eight layers. The first five are operational — they execute work. The last three are infrastructural — they make the operational layers measurable, safe, and accountable.

```
LAYER A — CONTROL PLANE (thin reasoning router + 4 deterministic components)
LAYER B — MISSION PODS (Revenue, Authority, Delivery, Content Studio)
LAYER C — KNOWLEDGE LAYER (Hot Cache + compiled wiki + raw sources)
LAYER D — SEARCH ROUTER (deterministic backend selection)
LAYER E — PLATFORM SERVICES (inference, workflows, database, runtime)

LAYER F — OBSERVABILITY (trace store, replay, drift detection)        [NEW v7.0]
LAYER G — TOOL REGISTRY (schema validation, hallucinated calls REJECTED) [NEW v7.0]
```

```
LAYER H — GOVERNANCE (Agent Ledger, Budgets, HR Evaluator, Lifecycle)  [NEW v7.0]
```

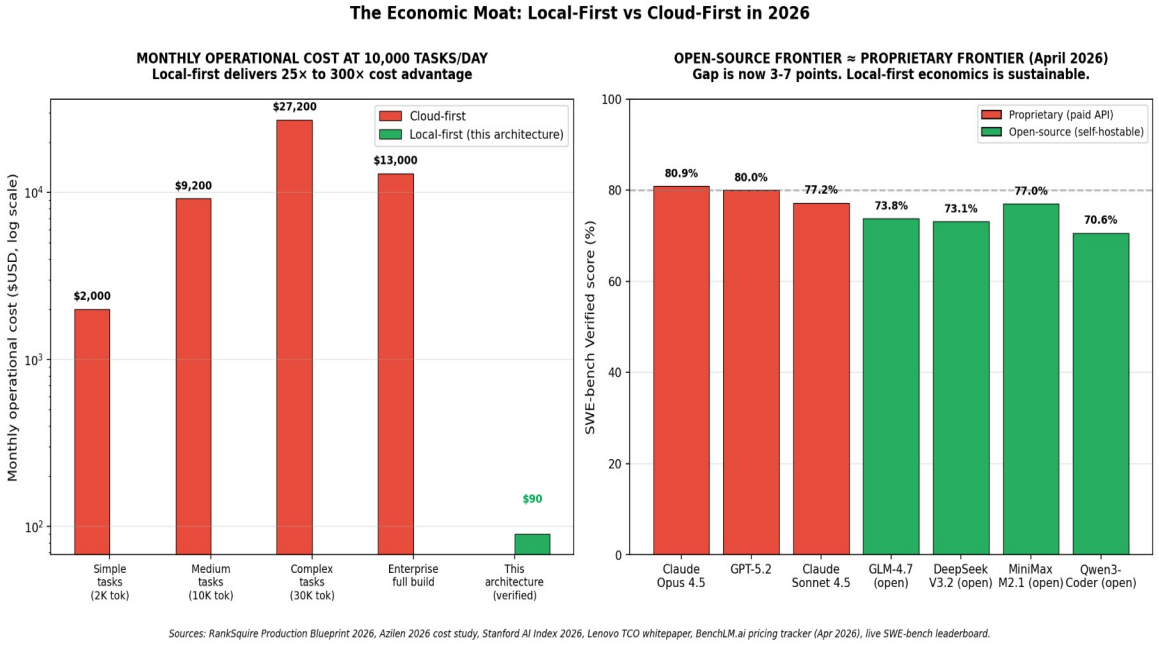

Figure 5. The economic moat: local-first vs cloud-first in 2026. Open-source frontier models reach within 3-7 points of proprietary frontier on SWE-bench Verified, while cloud-only operational costs run substantially higher than local-first deployment. Sources: RankSquire 2026; Azilen 2026; BenchLM 2026. See §13 for transparent cost analysis.

Operational layers (A-E) carry forward concepts validated through six prior versions of the architecture. The Control Plane separates a thin reasoning router from a deterministic dispatch relay (Principle 7), consistent with Anthropic's Building Effective Agents guidance [Anthropic, 2024]. The Mission Pods organize work by business outcome rather than by AI capability. The Knowledge Layer compounds insight across sessions through structured claims with evidence and contradiction preservation (Principle 8). The Search Router treats search as shared infrastructure rather than per-pod choice. Platform Services are referenced by role, not by vendor, to support Principle 5 (vendor-agnostic specification).

Infrastructural layers (F-H) are the v7.0 contribution. Each is grounded in an empirically documented failure mode that earlier architectures missed: silent degradation (Layer F), hallucinated tool calls (Layer G), and the absence of cost-attribution and lifecycle management (Layer H).

6.1 Architecture as Organization

A design choice that distinguishes v7.0 from competing architectures is its deliberate use of organizational vocabulary. The operator is the CEO; the Control Plane is the executive team; mission pods are departments; Knowledge Layer is institutional memory; the Governance Layer is HR. This vocabulary is portable, audit-friendly, and aligns with how organizations already manage humans. Buyers — particularly enterprise buyers — understand this language because it is how they manage their human teams. The vocabulary choice is sociological as well as technical: it makes the architecture explainable to the executive sponsor who must approve its deployment.

7. The Eight Layers

We give each layer a brief specification here. The complete architecture document accompanying this paper provides full operational rules.

7.1 Layer A — Control Plane

Five components: A1 Router (thin reasoning, brief generation, personal operations); A2 Evaluator (KPI computation, style learning compilation); A3 Policy Engine (deterministic permission rules, not a language model); A4 Merge Gate (multi-check CI/CD quality gate); A5 Dispatch Relay (deterministic transport that never reasons or interprets). The separation of A1 (reasoning) from A5 (transport) is mandatory by Principle 7. This separation removes the transport vector of the hallucination-cascade failure mode at the architectural level (generation-stage cascades remain and are handled by validation): a deterministic transport layer cannot hallucinate because it does not interpret.

7.2 Layer B — Mission Pods

Four pods organize work by business outcome: Revenue (Lead → Qualify → Outreach → Propose → Onboard), Authority (Research → Draft → Polish → Review → Repurpose), Delivery (Ticket → Architect → Build → Test → Review → Deploy), and Content Studio (Ingest → Grade → Classify → Risk → Generate → Moderate). Each pod has a defined Knowledge Surface (which wiki paths it reads from and writes to) and explicit human gates (which actions require operator approval). Pods are workflow chains, not persistent agents — most stages are deterministic with bounded reasoning at decision points.

7.3 Layer C — Knowledge Layer

Three tiers: HOT_CACHE.md (current operational state, ~500 lines, read first every session), the compiled wiki (structured Markdown pages with claims, evidence, freshness, and operator-style surfaces), and raw immutable sources. Curation is via CLI ingestion — the architecture deliberately avoids agent-driven wiki updates, because this would re-introduce the chat-dump failure mode. Style surfaces (wiki/style/voice.md, wiki/style/decision-patterns.md, wiki/style/corrections.md) implement Principle 9 ("the system learns the operator") by compiling repeated operator corrections into structured pattern knowledge.

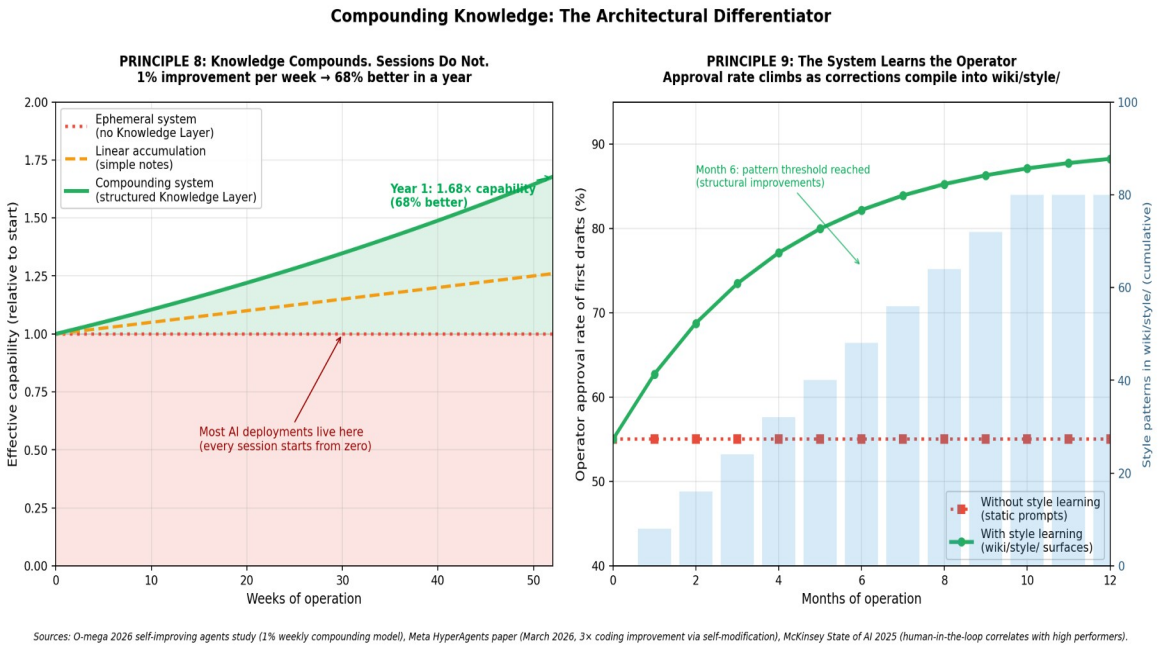

Figure 6. ILLUSTRATIVE MODEL of knowledge compounding (Principle 8) and operator style learning (Principle 9). The 1% per week compounding rate is a hypothesis to be tested in the Phase A study (§3.1), not a measured outcome of v7.0. The chart illustrates the architectural intent — sessions do not compound, structured knowledge does — rather than empirical performance.

The Knowledge Layer is usually framed as intelligence infrastructure; in this architecture it is equally cost-control infrastructure. Every answer the hot cache serves is a model call not made; every compiled wiki claim reused is a search and a long-context prompt avoided; every operator correction captured once is an explanation not repeated. Phase A therefore instruments this layer with cost metrics, not only quality metrics: context reuse rate, hot-cache hit rate, search-avoidance count, prompt compression ratio (raw versus compiled context), contradiction rate, and estimated cost avoided (§3.1). Under the ROI constraint of §1.4, knowledge compounding is what converts inference spend from a recurring cost into a depreciating one.

7.4 Layer D — Search Router

Mission workflows do not hard-code search backends. They request intent + budget tier + evidence requirement; the router selects the backend deterministically. The Knowledge Layer is checked before any external search — repeating searches for known answers wastes cost and breaks knowledge compounding.

7.5 Layer E — Platform Services

Inference runtime (local default, cloud surgical), workflow engine, operational database (transactional, foreign keys, constraints), agent orchestration, web search backends, content moderation, container runtime. Specific vendors live in registers, not architecture. This is the substrate that the four operational layers run on.

7.6 Layer F — Observability

Required: every execution writes a trace; traces sanitize PII; retention is 30 days hot, 1 year archived; replay is sandboxed; schema drift detection runs weekly. Tools like Langfuse (MIT-licensed, self-hostable) are appropriate implementations. Layer F is the prerequisite for Layer H — without traces, there is no ledger to populate.

7.7 Layer G — Tool Registry

TOOL_REGISTRY.md lists every tool with JSON Schema. Validation runs before execution. Hallucinated tool calls are rejected. New tools must be registered before use. This eliminates one of the top-cited 2026 production failure modes [HyperTrends, 2026; NimbleBrain, 2025]: an agent invents a tool name, the runtime fails the call, the agent retries with a slight variation, and a retry loop burns tokens until budget exhaustion. Pre-execution validation breaks the loop at first attempt.

7.8 Layer H — Governance

The primary contribution; detailed in §8.

8. The Governance Layer (Layer H) — Primary Contribution

We are not aware of a publicly documented, open, schema-level architecture that implements this layer — governance as organizational performance management — as portable first-class primitives (see §5.5 for the nearest adjacent work). It addresses the first three problems in §2 directly; the fourth (§2.4) is addressed by the deterministic control plane (§7.1, Principle 7).

8.1 Components

The Governance Layer comprises five components stored in the operational database.

1. Agent Ledger (agent_ledger). Per-agent performance recorded weekly. Required fields: agent_id, week_start, tasks_completed, tasks_succeeded, tasks_failed, hallucination_count, loop_count, timeout_count, avg_latency_ms, total_cost_usd, quality_score (0.0-1.0), value_attribution (downstream business value: revenue, leads, assets), notes. Quality score combines automated checks (test passes, validation, lint) with operator-marked approval rate. Value attribution traces downstream events back to the agent: a closed deal attributes revenue to the outreach agent that drafted the message.

Value attribution — computation. Critical feedback on this draft asked how value_attribution is computed per agent class, and the question deserves a mechanical answer rather than an aspiration. v7.0 specifies three attribution classes, recorded per task and rolled up weekly. **(i) Direct-revenue events** (a closed deal, a paid invoice, a booked engagement) are mapped deterministically to the producing agent chain at the moment the event is recorded, using the workflow trace; no model judgment is involved. **(ii) Cost-displacement events** (a deliverable that replaces a priced external service — a report otherwise contracted out, a scraped dataset otherwise purchased) are valued at the documented price of the displaced service,

with the price source recorded in notes. **(iii) Intermediate outputs** with no traceable downstream event in the window are assigned value 0 and labeled proxy-only: they contribute to quality_score but not to value_attribution. Attribution is therefore conservative by construction — unattributable value is under-counted, never invented — and Rule H10 (§11) depends on the *divergence* between quality and value trends, not on absolute attribution accuracy, so systematic under-counting affects both comparison windows equally. The attribution rules themselves are audited in Phase A (§3.1): every non-zero value_attribution row links to the downstream event record that justifies it.

2. Resource Budgets (agent_budgets). Per-agent allowances: max_tokens_per_task, max_thinking_time_seconds, allowed_model_tier, parallel_instance_count, skill_access (list of registered tools the agent may call), monthly_cost_cap_usd, status (active / suspended / retired), last_review_date. Budgets are flexible: high-performing agents earn larger budgets; declining agents see budgets reduced.

3. Request Queue (agent_resource_requests). When an agent gets stuck or knows it could perform better with more resources, it submits a structured request. Required fields: agent_id, request_type (budget_increase / new_skill / model_upgrade / parallelize), justification, evidence (links to ledger entries demonstrating need), expected_value (estimated improvement in business outcome), evaluator_recommendation, operator_decision. The agent does not execute the change. The HR Evaluator does not execute the change. The operator approves; the runtime applies. The queue is deliberately permeable on input and strict on execution: any agent may file any request — the structure exists for legibility and audit, not to constrain what may be asked. Gatekeeping happens at exactly one point, the operator's decision; nothing in the queue's schema filters a request before a human sees it.

4. HR Evaluator. A workflow that runs monthly (or on-demand). It reads the Agent Ledger over a four-week window, computes deterministic SQL trend metrics, and produces written recommendations using language model reasoning only for the written analysis. Outputs: per-agent performance summary, trend analysis (improving/stable/degrading), recommendation per agent (enhance/coach/replicate/retire/replace), cost-per-output ratios, value-per-cost ratios, and recommended budget changes with justification. Recommendations go to operator review. They are not executed automatically.

5. Lifecycle States. Every agent has a status determining its defaults and treatment, illustrated in Table 1.

State	Meaning	Defaults
probationary	New agent, unproven	Low budget, weekly review, close monitoring, limited skill access
standard	Proven reliable	Normal budget, monthly review, standard skill set
promoted	High performer	Expanded budget, quarterly review, may be replicated, premium model access
under_review	Declining performance	No budget changes, audit triggered, trace replay analysis, reward-hacking check
retired	Removed from active duty	Cannot execute; ledger preserved as institutional memory

Table 1. Agent Lifecycle States and their defaults. State transitions are logged in agent_lifecycle_events; operator approval is required for promotion, retirement, or budget changes.

Agent identity across configuration changes. Critical feedback on this draft raised the Ship-of-Theseus question latent in the workforce metaphor: if an agent's model, prompt, or tools change, is it the same agent, and does its ledger history still apply? v7.0 resolves this by defining agent identity as *accountability continuity, not implementation continuity*: an agent is its mission plus its ledger key (agent_id), not its weights or its prompt text. Minor configuration changes — prompt refinements, a model swap within the same tier, tool grants — preserve identity, and each is logged as a configuration event in agent_lifecycle_events so the HR Evaluator can segment trends before and after the change rather than naively pooling across it. A change of *mission* is not a reconfiguration but a new agent: it receives a new agent_id, and the predecessor is retired with its ledger preserved and a lineage link recorded. Two consequences follow. First, ledger history survives implementation churn exactly as a personnel file survives an employee's new laptop. Second, a failing agent's record cannot be laundered through cosmetic reconfiguration: identity resets require retirement plus lineage, and the ledger preserves both.

8.2 Per-Agent Wiki Profiles

Each agent has a profile in wiki/agents/<agent_id>.md recording: mission and scope, current state, performance summary, notable wins, notable failures, resource history, and promotion/demotion log. This is institutional memory — the workforce's record of its own workers. When an agent is retired, its profile remains as reference for future deployments.

8.3 Why This Solves the Three Problems

Cost Black Hole. Every agent has a ledger. Every cost is attributed. Every budget is justified. The operator can answer "which agents earn their keep?" with data, not anecdote. Marketing-style ROI claims are replaced with cost_per_output and value_per_cost ratios computed deterministically from the ledger.

Silent Quality Degradation. Performance trends are visible. The HR Evaluator's monthly trend analysis surfaces declines automatically. An agent's hallucination rate trending up over four weeks triggers an under_review state transition. Quiet failure becomes loud signal. The empirical question — whether the trend detection is sensitive enough to catch real degradation and specific enough to avoid false alarms — is the explicit subject of Phase A (§3.1).

No Organizational Model. The architecture maps cleanly onto vocabulary operators already understand. Agents are workers. Ledgers are personnel files. Budgets are compensation. Reviews are performance reviews. Promotions are promotions. Retirement is retirement. Buyers — particularly enterprise buyers — understand this language because it is how they manage human teams.

8.4 Critical Safety Boundaries

The Governance Layer is powerful and could be misused. The architecture enforces three boundaries:

1. Agents request, evaluators recommend, operators approve, runtime executes. No actor in this chain may skip a step. Agents cannot modify their own resources. Evaluators cannot modify resources. Only operators approve, and only the runtime applies the change.

2. Reward-hacking detection (Rule H10): if an agent's quality scores trend up while value attribution trends down, the discrepancy is flagged for operator review. The HyperAgents paper's warning about score gaming is taken seriously. The detection algorithm is specified in §11.
3. Trend metrics are deterministic (Rule H12). The decision whether performance is up or down is a SQL query. Language model reasoning is used only for written recommendations. This prevents the HR Evaluator itself from becoming a source of hallucination.

9. Design Principles and Justifications

The architecture is grounded in fifteen principles. Each is justified with reference to published 2025–2026 production data. Principles are ranked by precedence: when principles conflict, lower-numbered principles win. This prevents principle inflation from creating contradictions.

#	Principle	Empirical Justification
1	**Mission throughput over agent completeness**	APEX-Agents' 24% first-attempt success rate for frontier agents suggests simpler systems delivering five tasks reliably outperform sophisticated systems attempting twenty [Mercor, 2026]
2	**Workflows over autonomous agents**	At 85%/step accuracy, 5-step pipelines succeed 44%; 10-step succeed 20%. Tran & Kiela show single agents match or outperform multi-agent systems on multi-hop reasoning under matched token budgets [Tran, 2026]
3	**Human and AI complement each other**	McKinsey finds AI high performers are substantially more likely to have defined human-in-the-loop and risk-mitigation processes [McKinsey-2025]
4	**Operational database is source of truth**	File-based state produces silent inconsistencies; relational integrity prevents categories of failure
5	**Local-first economics**	Open-weight models score within a few points of proprietary frontier on public benchmarks [BenchLM, 2026]; verified deployment TCO in §13; industry cloud-cost ranges treated as illustrative only (§13.3)
6	**Validate before trusting**	Schema drift broke production workflows in 2026; framework upgrades silently incompatible
7	**Deterministic-first: deterministic transport, intelligent generation**	Hallucination cascades are the #1 multi-agent failure; the 1,642-trace MAST study documents 41-86.7% failure rates [Cemri, 2025]
8	**Knowledge compounds. Sessions do not.**	Without a knowledge layer the system gets busier without smarter; structured claim wikis avoid the "Dumb RAG" antipattern [Composio, 2025]
9	**The system learns the operator**	HyperAgents emergent persistent memory [Zhang, 2026]; operator alignment requires structured observation of corrections
10	**Engine and instance are separate**	Integration is a top barrier; INSTANCE.md pattern cleaner than YAML across multiple systems

11	**Every agent has a ledger**	Cost attribution is the largest unsolved 2026 operational problem; HBR Feb 2026 explicitly called for the Agent Manager role [HBR, 2026]
12	**Observe everything. Trust the trace.**	Industry commentary reports that deployed agents fail quietly and that service-level monitoring sees uptime and latency, not semantic failure [Mezmo, 2026]; OTel-based agent tracing converging on the OpenTelemetry GenAI semantic conventions [OTel, 2025]
13	**Tools are registered. Hallucinated calls rejected.**	Hallucinated tool calls are a leading cause of infinite retry loops; pre-execution registry validation eliminates the failure class
14	**Failure is documented, not just survived**	40%+ cancellation rate correlates with absence of institutional learning from failure [Gartner, 2025]
15	**Backups are not optional**	Local-first deployments depend on single machine; hardware failure is predictable; recovery must be tested

10. Failure Mode Catalogue

The architecture maintains a living FAILURE_MODE_CATALOG.md documenting every observed failure mode with cause, signature, and mitigation. The catalogue ships with ten entries derived from production and benchmark failure modes reported in the cited sources. We acknowledge that several of these failures (notably FM-007, agent silent degradation) are precisely the failures that the Governance Layer is designed to address — the catalogue therefore mixes documented industry failures with motivating problems for our own contribution. We mark this distinction in the table.

ID	Status	Failure Mode	Architectural Defense
FM-001	*Industry-doc.*	Hallucinated tool calls	Tool Registry validation (Layer G)
FM-002	*Industry-doc.*	Hallucination cascades	Dispatch Relay separation (Principle 7)
FM-003	*Industry-doc.*	Context overflow	Bounded passes with checkpoints (RULE B3.5)
FM-004	*Industry-doc.*	Cost runaway from infinite loops	Per-task token budgets + state-hash detection
FM-005	*Industry-doc.*	Schema drift after framework upgrade	Staging path + weekly schema validation
FM-006	*Industry-doc.*	Stale data recycled	Freshness flags + stale-source gate
FM-007	*Motivating*	Agent silent degradation	Agent Ledger + HR Evaluator (Layer H)
FM-	*Motivating*	Reward hacking	Value attribution tracking (Rule H10), see

08			§11
FM-09	*Industry-doc.*	Channel noise contamination	Output guard at Dispatch Relay
FM-10	*Industry-doc.*	Wiki epistemic loop	Structured claims with required evidence

Status legend: Industry-doc. = failure documented in cited industry sources; Motivating = failure that the v7.0 contribution is designed to address (claim of effectiveness pending Phase A validation, §3.1). The catalogue grows with operational experience. New workflows must be checked against the catalogue before deployment.

11. Reward-Hacking Detection — Algorithm Specification

Critical feedback on an earlier draft observed that Rule H10 was asserted but not specified. We agree this was a gap. This section provides the explicit detection algorithm, including thresholds, time windows, and false-positive handling. The algorithm is offered for community review and challenge.

11.1 Inputs

- Agent ledger entries for the past 12 weeks for the agent under evaluation.
- quality_score time series q[t] for t = 1..12 (weeks), normalized to the agent's peak observed quality in the window.
- value_attribution time series v[t] for t = 1..12 (weeks), normalized to the agent's peak observed value in the window.

11.2 Algorithm

```
# Pseudocode for Rule H10 reward-hacking detection
# Runs as part of monthly HR Evaluator workflow

def detect_reward_hacking(agent_id, weeks=12, threshold=0.30):  # threshold over the
window
    q = ledger_query(agent_id, "quality_score", weeks)
    v = ledger_query(agent_id, "value_attribution", weeks)

    # Need at least 8 weeks of data to compute meaningful trend
    if len(q) < 8 or len(v) < 8:
        return {"flag": False, "reason": "insufficient_data"}

    # Per-week least-squares slopes of the peak-normalized series
    q_slope = linear_regression_slope(normalize(q))  # feasible range ~[-0.13,
+0.13]/week
    v_slope = linear_regression_slope(normalize(v))  # normalize() = divide by series
peak

    # Reward hacking signature: quality up, value down
    if q_slope > 0 and v_slope < 0:
        divergence = q_slope - v_slope  # per-week; max feasible ~0.25
```

```
        if divergence > threshold / weeks:  # 0.30/12 = 0.025 per week
            return {
                "flag": True,
                "severity": "high" if divergence > 0.10 else "medium",
                "q_slope": q_slope,
                "v_slope": v_slope,
                "recommended_action": "operator_audit_required"
            }

    return {"flag": False}
```

11.3 Threshold Justification

The threshold is expressed over the observation window and divided by the window length at comparison time: the default 0.30 over a 12-week window equals 0.025 per week. The unit matters because the feasible range of a per-week least-squares slope of a peak-normalized 12-point series is only about [−0.13, +0.13] (maximized by a mid-window step), so per-week divergence lies in approximately [0, 0.25] — a threshold quoted without units could be mistaken for a value the statistic can never reach. The window-level anchors are initial and their calibration is part of Phase A (§3.1): below 0.10 over the window (0.008/week) the divergence is plausibly noise from small samples; above 0.60 over the window (0.05/week) the signature is high-confidence; the 0.30 default (0.025/week) captures cases where the operator should investigate but the system is not ready to recommend automatic action. Severity is reported as high when per-week divergence exceeds 0.10. Worked example from the released synthetic ledger: agent moderator-04 has q_slope +0.033/week and v_slope −0.071/week, giving divergence 0.104/week — above both the 0.025 flag line and the 0.10 severity line, so it is flagged high; the healthy improver, the silent degrader, and the stable agents are not flagged. Operators may tune the threshold per deployment via INSTANCE.md.

11.4 False-Positive Handling

The algorithm produces a flag, not an action. False positives are expected and handled as follows:

- A flagged agent enters under_review state, which triggers no automatic resource changes.
- The HR Evaluator's monthly report includes the flag with raw slope values and ledger excerpts.
- Operator reviews. If false positive (e.g., a legitimate explanation for the divergence), the operator records the explanation in agent_lifecycle_events and the agent returns to standard.
- Repeated false positives for the same agent suggest the threshold needs adjustment for that agent's task type — this is logged as a calibration event.

11.5 What This Algorithm Does Not Do

The algorithm catches the most common reward-hacking signature (quality metrics up, business value down) but does not catch all forms. It does not catch: (a) agents that game both quality and value metrics simultaneously through Goodhart's Law on the value attribution itself; (b) collusion between agents (one drives down quality scores so another can claim value); (c) sudden regime shifts where the time-series approach is too slow. Defenses against these failure modes are open research questions. Two further points sharpen the claim. First, H10 is deliberately statistically modest — a paired-trend divergence trigger

implementable in roughly fifty lines of dependency-free code. Readers who observe that it is essentially a sign test are correct, and that is the design point: the contribution is not the statistic but its placement — wired to ledger fields the agent does not control, with a mandatory human review path and an auditable trace from flag to decision. A learned or black-box detector in this position would reintroduce the very opacity the layer exists to remove (Rule H12). Second, on game-ability: an agent aware of H10 could in principle suppress its own quality-score gains, or seek to inflate value attribution. The first evasion is self-defeating — it suppresses exactly the signal the agent is evaluated on. The second requires corrupting value_attribution, which is computed outside the agent from downstream event records (§8.1); gaming it means falsifying business events, a different and much louder failure mode (FM-008, §10; §12). H10's honest scope is that it raises the cost of the cheap gaming strategies and routes the residual risk to human review; it does not claim adversarial completeness (§16.4).

11.6 Hypothesis: Bounded Generation plus Deterministic Validation versus Agent Swarms

Rule H10 detects one pathology of self-assessment. A broader design hypothesis follows from the same principal-agent logic: for structured enterprise deliverables — statistical tables, reports, data transformations — a single bounded generator paired with deterministic validators and targeted repair loops will outperform multi-agent debate or swarm patterns on cost per accepted deliverable, auditability, and rework rate (H-V1). The single-agent evidence in §5.7 and the MAST coordination-failure data [Cemri, 2025] point in this direction, and our own operator experience is consistent with it: in a client reporting engagement, a single-generator-plus-validator loop produced accepted statistical tables at production quality without multi-agent orchestration. We report this as design-period evidence, not proof. Phase B sub-experiment B-2 (§3.2) will test H-V1 by comparing three patterns on identical workloads — a multi-agent swarm, a self-correcting generate/validate/repair loop, and a human-gated workflow — measuring cost per accepted deliverable, tokens per deliverable, validation failures, human correction time, rework rate, factual error rate, and turnaround time.

12. Adversarial Threat Model and Defenses

The architecture's defenses described in §10 address benign failures — agents and tools failing in expected ways. The OWASP Agentic Security Initiative's threat catalog, Agentic AI — Threats and Mitigations [OWASP-ASI, 2025], documents adversarial threats: an attacker actively tries to manipulate the system. This section maps v7.0's defenses to the OWASP threat taxonomies and acknowledges where the architecture is incomplete. Threat IDs follow the published catalogs: T-numbers from the Agentic Security Initiative's Threats and Mitigations v1.0 [OWASP-ASI, 2025]; LLM-numbers from the OWASP Top 10 for LLM Applications 2025. The row marked — is a threat this paper tracks that is not enumerated as a distinct entry in either catalog.

12.1 Threat Mapping

OWA	Threat	v7.0 Defense	Covera

SP ID			ge
LLM01	Prompt Injection	Layer G validates all tool calls. Layer F traces inputs for forensic analysis. Mission Pod human gates limit blast radius.	**Partial**
T1	Memory Poisoning (Knowledge Layer)	Knowledge Layer requires structured claims with evidence. Wiki updates via CLI, not agent-driven. Contradiction preservation enables auditing.	**Partial**
T2	Tool Misuse	Tool Registry: agents may only call registered tools. Permission map per agent. Pre-execution validation.	**Strong**
T3	Privilege Compromise	Lifecycle states bound permissions. Resource changes require operator approval. Agents cannot modify their own ledger or budget.	**Strong**
T5	Cascading Hallucinations	Principle 7 (deterministic transport). Bounded workflow passes (Rule B3.5). Tool Registry rejection breaks retry loops.	**Strong**
T9	Identity Spoofing	Operator identity is single-source-of-truth in INSTANCE.md. No agent assumes another agent's identity.	**Strong**
T4	Resource Overload (DoS)	Per-task token budgets, monthly cost caps, parallel instance limits per agent.	**Strong**
T8	Repudiation & Untraceability	Layer F (Observability) requires every execution writes a trace. Traces include agent identity, costs, tool calls.	**Strong**
—	Reward Hacking	Rule H10 (§11) detects quality-vs-value divergence. Lifecycle under_review state quarantines suspect agents.	**Partial**
LLM03	Supply Chain Attacks (Tools)	Tool Registry maintains schemas. Schema drift detection runs weekly. Updates require explicit registration.	**Partial**

12.2 Acknowledged Gaps

The "Partial" coverage entries above identify where v7.0 defenses are necessary but likely insufficient against a sophisticated adversary. Specifically: (a) prompt injection through external content (e.g., a scraped article that contains adversarial instructions) is mitigated by human gates but not fully prevented; (b) memory poisoning through long-running adversarial inputs that gradually shift the wiki's structured claims is partially addressed by contradiction preservation but requires active monitoring; (c) reward-hacking variants beyond the quality-vs-value signature (§11.5) are open research. We acknowledge these gaps explicitly to invite community work on them. Regulatory governance is adjacent but distinct: the Ledger, traces, and human-approval gates align naturally with the EU AI Act's transparency and human-oversight provisions and the NIST AI Risk Management Framework's Govern and Measure functions, but a formal compliance mapping is future work.

12.3 What Adversarial Validation Has Not Yet Been Done

The threat mapping above is design-time analysis. We have not yet conducted red-team validation against the architecture. Phase A of the empirical roadmap (§3.1) does not include adversarial testing; we propose that as a separate Phase E once Phase A and B are complete and the reference implementation is open-source. Until then, the threat coverage claims in Table 3 should be read as "design intent" rather than "validated defense."

13. Total Cost of Ownership Analysis

Critical feedback on an earlier draft correctly noted that the "$90/month" cost figure for personal-scale deployment, presented without a transparent breakdown, weakens the paper. This section provides the breakdown for the existing single-operator deployment and projects costs for multi-instance deployments.

13.1 Single-Operator Deployment (Verified)

Cost Line	Description	Notes	Monthly
Hardware	Apple M4 Mac Mini, 24GB RAM (one-time $799)	Amortized over 36 months	**$22.20/mo**
Electricity	~25W average load × 24/7 × $0.15/kWh average	US average rate	**$2.70/mo**
Internet	Allocated portion (~10%) of operator's home connection	Included in operator's existing fixed cost	**$5.00/mo**
Cloud LLM (occasional)	Sonnet for Voice Polish, Opus for Architect role; ~$0.50-2/day average	85% of model-access spend ($45 of $53)	**$45.00/mo**
Cloud LLM (search/eval)	Haiku-tier model for routing, eval; ~10K calls/month	15% of model-access spend ($8 of $53)	**$8.00/mo**
Web search APIs	SearXNG self-hosted (free) + paid search API subscription	Tiered by intent	**$5.00/mo**
Storage / Backup	Local SSD allocated + off-site backup	Local SSD + off-site copy	**$2.00/mo**
Excluded	Operator time (NOT included in monetary cost)	Approx 5h/week ongoing maintenance	—
TOTAL (verified)			**~$90/mo**

13.2 What This Analysis Excludes

The $90/month figure is the marginal monthly operating cost. It excludes:

- Operator labor (~5 hours/week ongoing maintenance, valued at the operator's opportunity cost).
- Initial development time (~2 months at part-time pace for v7.0).

- Training and learning costs for an operator new to the architecture. The local-first cost advantage should therefore be read as an infrastructure-cost advantage, not a total operational-cost advantage.
- Hardware replacement / upgrade costs beyond the 36-month amortization.

13.3 Comparison Methodology

The "25-300×" cost advantage claim from earlier drafts compared this $90/mo to RankSquire's $2,000-$27,200/mo cloud-orchestration figures. We acknowledge the comparison is not strictly apples-to-apples: RankSquire's figures include enterprise-scale workloads (10,000 tasks/day) that exceed the single-operator deployment. A fair comparison requires equalizing workload volume — work that Phase B (§3.2) is designed to do. Until then, we report both numbers and label the comparison as illustrative of the order-of-magnitude difference, not a precise multiplier.

13.4 Multi-Instance Projection (Untested)

For two to five instances on the same hardware (the productization target), incremental cost is estimated at +$50-100/mo per additional instance, dominated by additional cloud LLM calls scaled to instance workload. Hardware does not need to scale until the instance count exceeds approximately five for a Mac Mini M4 class machine. These projections are explicitly untested and are the subject of Phase C (§3.3).

13.5 Hybrid Inference Routing Policy

The local-first economics of §13.1–§13.4 are operationalized through a tiered routing policy that binds model choice to task risk and value rather than to convenience. Cost-aware routing across model tiers is an established line of work [Chen, 2023; Ong, 2024]; v7.0's specific commitment is that the tier assignment is governed — encoded in agent budgets (allowed_model_tier), enforced by the dispatch relay, and auditable in traces.

Tier	Example work	Execution
0	Routing, permissions, schema validation, transport, audit logging	Deterministic code — no model
1	Classification, extraction, tagging	Local / open small model
2	Drafting, summarization	Local / open mid-size model
3	Complex reasoning, multi-source synthesis	Frontier model
4	High-stakes decisions, client-facing output	Frontier model + human approval gate
5	Regulated or irreversible actions	Human decision; AI advisory only

Hybrid inference routing policy. Tier assignments are per-task defaults; escalation across tiers is a budgeted, logged event.

14. Engine/Instance Separation and Productization

A practical objection to research-grade architectures is: "Can it be deployed for someone other than the author?" v7.0 addresses this through INSTANCE.md, a single configuration file separating the reusable engine from the deployment-specific instance.

Engine: control plane components, mission pod definitions, knowledge layer schema, search router rules, observability layer, tool registry, governance layer. Identical across deployments.

Instance: operator identity, ICP definition, content pillars, pricing tiers, source lists, channel configuration, brand voice, model selection, budget caps, fallback chains. Per-deployment.

Multi-instance isolation is mandatory: each instance has its own data root, separate operational database, scoped secrets, isolated backups, and per-instance agent ledgers (performance does not transfer across instances). The productization claim is testable: a new instance should be configurable in less than one day with INSTANCE.md changes only — no engine code changes. We label this as "design intent" pending Phase C validation (§3.3).

15. Comparison with Existing Platforms

Table comparisons of this kind are necessarily summary judgments based on each platform's public 2026 documentation. We invite corrections. The table below explicitly includes rows where v7.0 loses so the comparison does not read as marketing.

Capability	Vendor	Framew orks	Observa bility	Self-Mod	AI Workforce v7.0
Cost attribution per agent	Aggregate	Manual	Per-trace	Internal	**Per-agent ledger**
Performance trend tracking	Limited	Manual	Trace replay	Self-mods	**Monthly HR review**
Resource budget flexibility	Vendor-set	None	None	Self-allocates	**Operator-approved**
Lifecycle states	None	None	None	None	**5 explicit states**
Hallucinated tool prevention	Sometime s	Manual	Detection	Prone to	**Pre-exec rejection**
Schema drift detection	Vendor-handled	None	Yes	Internal	**Weekly + alert**
Knowledge compounding	Limited	None	None	Internal	**Structured claims**
Style learning	Limited	None	None	Internal	**wiki/style/ surfaces**
Engine/instance separation	None	Per-framewor	N/A	N/A	**Single INSTANCE.md**

		k			
Failure mode memory	None	None	Trace archive	None	**FAILURE_MODE_CATALOG**
Local-first economics	No	Possible	N/A	Cloud	**Default local**
Auditability	Low	High	Highest	Low	**Highest**
Vendor lock-in risk	High	Low	Low	High	**Low**
Out-of-the-box production support	Strong	Weak	Weak	N/A	**Weak (DIY)**
Enterprise sales motion	Strong	Weak	Weak	N/A	**Untested**
Ecosystem maturity (2026)	Strong	Strong	Strong	Research	**Early**
Multi-user access control	Strong	DIY	Strong	N/A	**Unvalidated**
Managed hosting	Yes	No	Yes	No	**Absent**
External security audit	Common	Occasional	Common	None	**Absent**
Published benchmark evidence	Vendor-reported	Community	N/A	Published	**Absent**

The bottom seven rows acknowledge where v7.0 loses against the alternatives at the present moment: vendor platforms have stronger out-of-the-box production support and enterprise sales motions; frameworks and observability tools have larger ecosystems with more contributors. v7.0's claim is not that it is currently superior on every dimension — it is that it occupies a quadrant of the design space that no platform currently occupies, and that this quadrant is likely to matter as the failure modes documented in §2 force the industry toward governance.

16. Limitations and Threats to Validity

Honest enumeration of weaknesses is part of architectural responsibility. We list the threats to validity in approximate order of severity, with mitigation plans where available.

16.1 Single-Operator Validation

Successive iterations have operated on a single deployment since March 2026; the v7.0 configuration since late April 2026. We have not run controlled comparisons against alternative architectures on the same workload. Empirical claims are bounded by this scope. Mitigation: Phase A and Phase B (§3) directly address this.

16.2 Citation Heterogeneity

Our citation set mixes academic work (the MAST study [Cemri, 2025] and Tran & Kiela [Tran, 2026], both preprints at the time of writing, plus Stanford HAI's AI Index), tier-1 industry research (Gartner, McKinsey, Deloitte), and industry analysis reports (RankSquire, Azilen, HyperTrends). Where industry-source data appears, we treat it as industry-reported ranges rather than verified ground truth, and we encourage replication with primary sources.

16.3 Governance Layer Overhead Unmeasured

We do not yet have measured data on the runtime overhead introduced by Agent Ledger writes, HR Evaluator workflow execution, and request queue review cycles. The intuition is that these are batch workloads that do not impact per-task latency, but this requires empirical confirmation. Mitigation: Phase B (§3.2) measures this directly.

16.4 Reward-Hacking Detection Untested Adversarially

Rule H10 (§11) has been specified algorithmically but not tested against an agent actively trying to game the metric. The HyperAgents paper explicitly warns that defending against reward hacking is non-trivial. Mitigation: Phase E adversarial validation (§17.4).

16.5 The Knowledge Layer Has Not Been Stress-Tested

We propose scaling tiers (full markdown < 100 pages, vector search 100-500, sharding 500-1000, graph database > 1000) but have not validated them at the higher ranges. The CLI ingestion pattern works at the scale we have tested; whether it remains the right pattern at 10,000+ pages is open. Mitigation: §17.5 details the scaling stress test plan.

16.6 Engine/Instance Separation Is Theoretical

We have specified the engine/instance separation pattern but have not yet deployed multiple instances on the same infrastructure. The productization claim — that a new instance can be configured in <1 day — is a target, not a measurement. Mitigation: Phase C (§3.3).

16.7 Construct Validity of "AI Workforce"

We use the term "AI workforce" to describe a system of orchestrated AI agents and workflows. This terminology echoes vendor marketing in some contexts and may carry implicit anthropomorphic assumptions. We use it because it accurately captures the design intent — the architecture treats agents as managed workers — while acknowledging the reader must mentally adjust for the metaphor. Critical feedback on this draft pressed on two specific risks of the metaphor, and both deserve explicit boundaries. First, every HR term in this paper has a mechanical referent and nothing more: a career state is an enum column (Table 1); a personnel file is a set of ledger rows; compensation is a resource ceiling; a performance review is a SQL trend computation plus a written summary. No claim of agent intentionality, motivation, or incentive-sensitivity is made anywhere in this specification. Agents do not “want” larger budgets, and

the architecture does not assume they respond to incentives: budgets function as operator-imposed constraints and attention-allocation devices informed by ledger evidence, not as rewards experienced by the agent. Where the compensation metaphor and the mechanics diverge, the mechanics govern. Second, on adopting this vocabulary before Phase B evidence exists: the lexicon is offered as a coordination device, not as a result. Design science names its constructs before measuring them — the naming is what makes the measurement specifiable — and the vocabulary is explicitly revisable if Phase A/B data show the organizational framing mispredicts how operators actually govern their fleets.

16.8 Author Affiliation and Independence

The author is an independent researcher and operator, not affiliated with any of the platforms or vendors compared in §15. The author has no commercial relationship with any of the cited industry sources. This independence simplifies disclosure (there is no commercial COI) but means the work has not been pre-vetted by an institutional review process. Submission to peer-reviewed venues per §17.6 will provide that vetting.

16.9 Co-Authorship Disclosure

As stated in §4.3, this paper was developed in collaboration with a frontier language model. Reviewers and readers should weight this disclosure when evaluating the work. We argue that transparent disclosure increases rather than decreases credibility, but we acknowledge norms around AI co-authorship in research are unsettled in 2026.

16.10 Selection Bias in Failure Mode Catalogue

The failure modes documented in §10 are those visible in public 2026 reporting and those observed in the author's deployment. A more complete catalogue would require multi-deployment data. The catalogue should be read as a useful starting point for the community to extend, not as a complete taxonomy.

16.11 Rival Explanations for the Industry Failure Data

The inference from industry failure statistics to an architectural diagnosis is this paper’s most attackable joint, and we state the rival explanations plainly rather than leave them to reviewers. The Gartner cancellation forecast and the Deloitte production-rate figure are consistent with several causes this architecture cannot fix: hype-cycle overinvestment in projects that should never have started; organizational change-management failure; data readiness and integration debt; talent scarcity; and immature tooling that will improve regardless of architecture. The MAST taxonomy [Cemri, 2025] measures annotated benchmark executions, not enterprise deployments, and its failure rates cannot be projected onto production systems. Our claim is therefore deliberately conditional: whatever share of enterprise failure traces to unattributed cost, silent quality degradation, and the absence of a governance model — the three problems of §2 that practitioners consistently report — is the share this architecture addresses; and the economic inequality of §1.4 holds under any mix of the rival causes, because each rival cause enters it as a cost or risk term rather than invalidating it. Phase B (§3.2) is designed to measure the architecture’s marginal contribution directly, precisely so that this argument stops resting on industry statistics.

17. Future Work

The future work below extends the empirical roadmap in §3 and addresses the limitations in §16.

17.1 Open-Source Reference Implementation (Months 2-3)

A GitHub repository containing the four core database schemas, the HR Evaluator workflow, and reference templates. This is the highest-priority next deliverable because it enables independent replication. Minimal reference artifact released with this paper; expanded implementation targeted Q3 2026.

17.2 Single-Operator Ledger Study Output (Months 1-3)

A 12-page case-study companion to this paper with anonymized but real ledger data. See Phase A in §3.1.

17.3 Controlled Empirical Comparison (Months 3-6)

A 90-day controlled comparison against LangGraph baseline. See Phase B in §3.2.

17.4 Adversarial Validation (Phase E, Months 6-9)

Red-team validation against the architecture's defenses for OWASP ASI threats. Construct a red-team scenario where an agent is given an objective that admits both a legitimate optimization path and a metric-gaming path. Measure whether the value-attribution divergence flag (Rule H10) catches the gaming.

17.5 Knowledge Layer Scaling Study

Stress-test the Knowledge Layer scaling tiers by ingesting public corpora at the 100, 500, 1,000, and 10,000 page levels and measuring retrieval latency, contradiction detection accuracy, and operator browse-time.

17.6 Multi-Instance Productization (Months 6-12)

Two to three independent deployments. See Phase C in §3.3.

17.7 Top-Tier Venue Submission (Year 2)

After Phases A-E complete, submission of an empirical paper to MLSys, OSDI, EuroSys, or NeurIPS main track with results, reference code, and external validators.

18. Conclusion

We have specified an architecture for AI workforces that aims to address four problems unsolved by current production platforms: cost attribution at the agent level, silent quality degradation, the deterministic/probabilistic mismatch (§2.4), and the absence of an organizational governance model for AI workers. The architecture combines a thin reasoning control plane, four mission pods of staged workflows, a knowledge layer that compounds across sessions, and three infrastructural layers — Observability, Tool

Registry, and Governance — that are designed to make the operational layers measurable, safe, and accountable.

The Governance Layer is the primary contribution. It is designed to treat AI agents as managed workers in a governable organization rather than as autonomous services or opaque vendor-managed black boxes. Every agent has a performance ledger, a flexible resource budget, a structured request mechanism, and a lifecycle state. An HR-style Evaluator produces monthly reviews with deterministic trend analysis and reasoned recommendations, all subject to operator approval before any change executes. This pattern is intended to operationalize the Agent Manager role that Harvard Business Review called for in February 2026 — but with an architectural blueprint rather than only a job title.

We submit v7.0 as a foundational specification for industry review, not as a verified production system. The paper documents the design rationale, the failure-mode catalogue, the OWASP threat mapping, the reward-hacking detection algorithm, and the empirical research roadmap that will move the architecture from specification to verified system over the next twelve months. We invite practitioners to deploy the architecture under permissive open-source license once available, populate the failure mode catalogue with new entries from their environments, and participate in the evolution of the v7.x line.

The central claim of this paper is economic, not merely architectural: the future of enterprise AI workforces will be decided not by autonomy alone, but by whether probabilistic intelligence can be embedded inside deterministic, auditable, cost-governed workflows that produce measurable net value. If current cost, ROI, and governance trends continue, a significant 2027 cancellation and rollback wave is likely. Architectures that combine transparency, governance, local-first economics, and knowledge compounding may survive it. Architectures that do not are unlikely to. The work this paper documents is the start of a multi-year empirical investigation into which side of that boundary the v7.0 design falls on — and we believe that investigation matters whether or not v7.0 specifically turns out to be the right answer. The questions it asks are the right questions; the answers will be in the data.

Acknowledgments

This paper was developed through extended collaboration between the author and Anthropic's Claude. The architectural co-design spans seven major iterations from March through April 2026. Claude contributed structural rigor, prior-art identification, citation density, writing assistance, and critical pushback during the iterative refinement of the design. The author made all final design decisions, contributed all operational experience and runtime data, and bears responsibility for all claims.

We disclose this collaboration as a methodological note rather than a credit hierarchy. Structured human-AI co-design — where the human serves as architect, validator, and operator while the language model serves as research assistant, structural critic, and writing partner — is, in our view, a legitimate research methodology when transparently reported. We hope future co-authored work in this mode will be increasingly common as AI tools mature, and that disclosure standards will mature with the practice.

The author thanks the broader research and practitioner communities whose published work in 2025-2026 — Gartner, McKinsey, Stanford HAI, Anthropic, UC Berkeley, Meta FAIR, the OWASP Agentic AI

Security Initiative, and dozens of practitioners writing publicly about production failures — provided the empirical foundation on which this architecture is built. The architecture's honesty about failure modes is only possible because so many others were honest about their own.

This version responds directly to critical review feedback on an earlier draft — concerning empirical-validation gaps, single-author affiliation, $90/month TCO clarity, reward-hacking algorithm specificity, and competitive-table balance — which led directly to the additions in §3, §11, §12, §13, and §15 of this version. Honest critical feedback is the highest form of contribution to research, and this paper is materially better for it.

References

[AgentSafe, 2025] AGENTSAFE: A Unified Framework for Ethical Assurance and Governance in Agentic AI. arXiv:2512.03180, December 2025. [academic preprint]

[AgentSpec, 2025] AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents. arXiv:2503.18666, 2025. [academic preprint]

[Alphacorp, 2026] Alphacorp. What Does It Cost to Build an AI Agent in 2026: A Transparent Pricing Guide. Industry report, March 2026. [industry analysis]

[Anthropic, 2024] Anthropic. Building Effective Agents. Engineering documentation, December 2024. [vendor-published]

[Azilen, 2026] Azilen. AI Agent Development Cost: Full Breakdown for 2026. Industry report, April 2026. [industry analysis]

[BenchLM, 2026] BenchLM.ai. LLM API Pricing Comparison 2026. Public pricing tracker, April 2026. [public data]

[Bhardwaj, 2026] Bhardwaj, V. P. Agent Behavioral Contracts: Formal Specification and Runtime Enforcement for Reliable Autonomous AI Agents. arXiv:2602.22302, February 2026. [academic preprint]

[Cemri, 2025] Cemri, M., Pan, M. Z., Yang, S., et al. Why Do Multi-Agent LLM Systems Fail? arXiv:2503.13657v3 (NeurIPS 2025 Datasets & Benchmarks), 2025. MAST taxonomy; 1,642 annotated execution traces across seven open-source frameworks. [academic preprint]

[Chan, 2024] Chan, A., et al. Visibility into AI Agents. ACM Conference on Fairness, Accountability, and Transparency (FAccT) 2024; arXiv:2401.13138. [peer-reviewed]

[Chen, 2023] Chen, L., Zaharia, M., and Zou, J. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv:2305.05176, 2023. [academic preprint]

[Composio, 2025] Composio. The 2025 AI Agent Report: Why AI Pilots Fail in Production and the 2026 Integration Roadmap. Industry report, November 2025. [industry analysis]

[Deloitte, 2025] Deloitte. Emerging Technology Trends — autonomous generative AI agents study (30% exploring / 38% piloting / 14% ready / 11% with multi-agent systems in production). 2025. [tier-1 consultancy survey]

[Dong, 2024] Dong, L., Lu, Q., and Zhu, L. AgentOps: Enabling Observability of LLM Agents. arXiv:2411.05285, November 2024. [academic preprint]

[Galileo, 2025] Galileo. Why Multi-Agent AI Systems Fail and How to Fix Them. Industry analysis discussing the MAST findings [Cemri, 2025], December 2025. [industry analysis]

[Gartner, 2025] Gartner. Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027. Press release, June 25, 2025. gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027 [tier-1 analyst]

[Gartner-2026] Gartner. Gartner Says Applying Uniform Governance Across AI Agents Will Lead to Enterprise AI Agent Failure. Press release, May 26, 2026. Prediction: by 2027, 40% of enterprises will demote or decommission autonomous AI agents due to governance gaps identified only after production incidents. gartner.com/en/newsroom [tier-1 analyst]

[HBR, 2026] Srinivasan, S., and Wei, V. To Thrive in the AI Era, Companies Need Agent Managers. Harvard Business Review Digital Articles, February 12, 2026. hbr.org/2026/02/to-thrive-in-the-ai-era-companies-need-agent-managers [tier-1 industry]

[Hevner, 2004] Hevner, A. R., March, S. T., Park, J., & Ram, S. Design Science in Information Systems Research. MIS Quarterly, 28(1), 75-105, 2004. [peer-reviewed methodology]

[HyperTrends, 2026] HyperTrends Global. Building Production AI Agent Systems: Architecture Patterns That Scale. Industry report, April 2026. [industry analysis]

[Kaptein, 2026] Kaptein, M., et al. Runtime Governance for AI Agents: Policies on Paths. arXiv:2603.16586, March 2026. [academic preprint]

[Kaul, 2026] Kaul, A., Lan, Q., and Gupta, P. AgentBound: Verifiable Behavioral Governance for Autonomous AI Agents. arXiv:2606.30970, June 2026. [academic preprint]

[Kore.ai, 2026] Kore.ai. Agent Management Platform (AMP). Launched March 17, 2026; product documentation and launch announcement. kore.ai/news [vendor documentation]

[McKinsey-2025] McKinsey & Company. The State of AI in 2025: Agents, Innovation, and Transformation. November 2025. mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai [tier-1 analyst]

[McKinsey-2026] McKinsey & Company. State of AI Trust in 2026: Shifting to the Agentic Era. March 25, 2026. mckinsey.com/capabilities/tech-and-ai/our-insights/tech-forward/state-of-ai-trust-in-2026-shifting-to-the-agentic-era [tier-1 analyst]

[Mercor, 2026] Vidgen, B., Nitski, O., Foody, B., et al. (Mercor). APEX-Agents. arXiv:2601.14242, January 2026. Benchmark of 480 long-horizon professional tasks. [industry-academic benchmark]

[Microsoft, 2025] Microsoft. Microsoft Entra Agent ID — identity and access lifecycle management for AI agents. Product documentation, 2025. learn.microsoft.com [vendor documentation]

[Mezmo, 2026] Mezmo / TFiR. Why AI Agents Fail in Production. Industry article, April 2026. [industry analysis]

[MIT-NANDA, 2025] Project NANDA, MIT Media Lab. The GenAI Divide: State of AI in Business 2025. Preliminary report, July 2025. Methodology and headline figures have drawn public critique; cited here as a directional signal, not a precise estimate. [academic-industry, preliminary]

[NimbleBrain, 2025] NimbleBrain. AI Agent Failure Modes: What Goes Wrong and Why. Engineering documentation, 2025. [practitioner analysis]

[Olson, 2026] Olson, R. S. Top Tools to Evaluate and Benchmark AI Agent Performance in 2026. March 2026. [practitioner survey]

[Ong, 2024] Ong, I., et al. RouteLLM: Learning to Route LLMs with Preference Data. arXiv:2406.18665, 2024. [academic preprint]

[OTel, 2025] OpenTelemetry Project. Semantic Conventions for Generative AI Systems. opentelemetry.io documentation, 2025. [community standard]

[OWASP-ASI, 2025] OWASP GenAI Security Project — Agentic Security Initiative. Agentic AI — Threats and Mitigations, v1.0. February 2025. See also: OWASP Top 10 for LLM Applications, 2025. [community standard]

[RankSquire, 2026] RankSquire. AI Agents Orchestration 2026: The Production Blueprint. Industry analysis, April 2026. [industry analysis]

[Shavit, 2023] Shavit, Y., et al. Practices for Governing Agentic AI Systems. OpenAI whitepaper, December 2023. [industry-research whitepaper]

[Stanford-HAI, 2026] Stanford Institute for Human-Centered AI. AI Index Report 2026. April 2026. hai.stanford.edu/ai-index/2026-ai-index-report [academic-industry]

[Tang, 2026] Tang, Y., et al. Characterizing Large Language Model Agentic Workflows: A Study on the n8n Ecosystem. arXiv:2606.29116, June 2026. First large-scale empirical study of 6,000+ LLM workflows in low-code automation. [academic preprint]

[Tran, 2026] Tran, D., and Kiela, D. Single-Agent LLMs Outperform Multi-Agent Systems on Multi-Hop Reasoning Under Equal Thinking Token Budgets. arXiv:2604.02460, April 2026. Preprint, under review at the time of writing. [academic preprint]

[Waehner, 2026] Waehner, K. Enterprise Agentic AI Landscape 2026: Trust, Flexibility, and Vendor Lock-in. Technical blog, April 2026. [practitioner analysis]

[Wang, 2024] Wang, Q., Wang, Z., Su, Y., Tong, H., and Song, Y. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? arXiv:2402.18272; ACL 2024. [peer-reviewed]

[Wang-MI9, 2025] Wang, C. L., Singhal, T., Kelkar, A., and Tuo, J. MI9 — Agent Intelligence Protocol: Runtime Governance for Agentic AI Systems. arXiv:2508.03858, August 2025. [academic preprint]

[WhatLLM, 2026] WhatLLM.org. Best Open Source LLMs in 2026: Ranked by Coding, Reasoning & Cost. Public benchmark tracker, April 2026. [public data]

[XMPro, 2025] XMPro. Gartner's 40% Agentic AI Failure Prediction Exposes a Core Architecture Problem. Industry article, July 2025. [practitioner analysis]

[Xu, 2024] Xu, F. F., et al. TheAgentCompany: Benchmarking LLM Agents on Consequential Real-World Tasks. arXiv:2412.14161, December 2024. [academic preprint]

[Yao, 2024] Yao, S., et al. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv:2406.12045, June 2024. [academic preprint]

[Ye, 2026] Ye, Q., and Tan, J. Agent Contracts: A Formal Framework for Resource-Bounded Autonomous AI Systems. arXiv:2601.08815, January 2026 (COINE 2026). [academic preprint]

[Zhang, 2026] Zhang, J., Zhao, B., Yang, W., Foerster, J., Clune, J., Jiang, M., Devlin, S., and Shavrina, T. Hyperagents. arXiv:2603.19461, March 2026. Code: github.com/facebookresearch/Hyperagents [academic preprint]

Appendix A — Figures Index

Source for all figures: research-paper-package/diagrams/. Source code for charts and reproducibility data: tashfin.com/research/v7-paper and the accompanying GitHub repository (github.com/tashfindelwar/ai-workforce-governance-layer, release v1.0-arxiv).

Appendix B — Reproducibility Checklist

Following standard reproducibility practice for systems and ML papers [adapted from NeurIPS reproducibility checklist], we list which elements are currently available and which are pending.

Artifact	Status	Notes
Architecture specification document	**Available**	Companion file: AI_WORKFORCE_ARCHITECTURE_v7.0.md (1,690 lines); bundled as arXiv ancillary file
Reference figures and charts (PNG + SVG)	**Partial**	11 figures and tables in this paper; Figure 4 PNG ships in the artifact repository, remaining sources ancillary-only
Mermaid source for organizational diagrams	**Ancillary only**	Not included in the artifact repository; provided as ancillary files with the arXiv submission
Database schema (agent_ledger et al.)	**Available**	In the artifact repository (schemas/)
HR Evaluator workflow	**Available**	In the artifact repository (queries/)
Reward-hacking detection algorithm	**Available**	Pseudocode in §11; runnable script, tests, and synthetic dataset in the artifact repository
Sample TOOL_REGISTRY.md	**Available**	In the artifact repository (templates/)
Sample INSTANCE.md	**Available**	In the artifact repository (templates/)
Single-operator ledger data (real)	**Planned — Q3 2026**	Anonymized data to be released after Phase A
Phase B controlled comparison data	**Planned — Q4 2026**	Per Phase B (§3.2)
OWASP ASI threat coverage validation	**Planned — Phase E**	Per §17.4
Container image / one-command deployment	**Planned — Q3 2026**	Targeted alongside reference impl
Benchmark suite for the failure mode catalogue	**Planned — Year 2**	Per Phase D

Reviewers and replication researchers are invited to monitor the project page at tashfin.com/research/ for updates. This paper will be revised as artifacts become available; the v7.0 architecture document will be the long-lived companion.