

# Toward a Governable Architecture for Agentic Enterprises

A domain independent framework spanning digital workflows and embodied systems.

**Author:** Sven D. Olensky

**Publisher:** Agency Collapse Publishing

**Date:** December 2025

**License:** Creative Commons Attribution 4.0 International, CC BY 4.0

**Copyright © 2025 Agency Collapse Publishing. All Rights Reserved.**

# Toward a Governable Architecture for Agentic Enterprises

## Zenodo Metadata Block

**Title:**

Toward a Governable Architecture for Agentic Enterprises - A domain independent framework spanning digital workflows and embodied systems

**Author:**

Sven D. Olensky

**Publisher:**

Agency Collapse Publishing

**Publication Type:**

Report / Research Paper

**Version:**

1.0 RELEASE

**Publication Date:**

2025-12-15

**Description:**

*Toward a Governable Architecture for Agentic Enterprises - A domain independent framework spanning digital workflows and embodied systems* proposes a domain independent architectural backbone for making agent autonomy scalable without making oversight structurally infeasible. It reframes governance as an enforceable runtime property rather than a primarily procedural afterthought, grounded in distributed systems constraints, human cognitive limits, and security first principles. The paper defines a small set of stable runtime primitives, including agent identity and authority, policy evaluation and enforcement, provenance and traceability, risk adaptive supervision, monitoring, and containment mechanisms for fault and misuse propagation. It then organizes these primitives into a portable reference architecture and illustrates operational consequences through worked scenarios of failure propagation and successful containment under partial failure and adversarial pressure.

**Keywords:**

Agentic AI, Agentic Enterprises, Governable Architecture, Runtime Governance, Control Planes, Distributed Policy Enforcement, Policy Mesh, Job Contracts, Domain Packs, World Models, Provenance, Attribution, Traceability, Monitoring, Supervision Modes, Containment, Zero Trust, NIST AI RMF, ISO/IEC 42001

**DOI:** [10.5281/zenodo.17943969](https://doi.org/10.5281/zenodo.17943969)

**License:** Creative Commons Attribution 4.0 International (CC BY 4.0)

**Language:** English

**Upload Type:** Report / White Paper

**Communities:**

- AI Safety and Security Research
- Agentic Systems Governance
- Enterprise Architecture Research

## Toward a Governable Architecture for Agentic Enterprises

### **Related Identifiers (optional):**

- Zero Trust Architecture
- Artificial Intelligence Risk Management Framework
- Artificial Intelligence Risk Management Framework Profile for Generative AI
- ISO/IEC 42001 Artificial Intelligence Management System
- ISO/IEC 23894 Artificial Intelligence Risk Management
- PROV Data Model
- Coalition for Content Provenance and Authenticity Specification
- Multi-Agentic System Threat Modeling Guide
- Agentic AI Threat Modeling Framework MAESTRO

## Contents

Purpose.....	7
Scope .....	7
Abstract .....	8
Introduction and Problem Definition.....	8
<b>Motivation</b> , scope, and research questions.....	9
<b>Key terms</b> and definitions .....	11
Paper organization .....	13
Theoretical Foundations: Cognition, Systems, and Enterprise Posture .....	14
<b>Human</b> cognitive limits .....	14
<b>Organizational</b> and distributed cognition.....	14
Distributed systems constraints and control planes.....	15
<b>Enterprise posture</b> and risk management expectations.....	16
From <b>Human-Centric Workflows</b> to Hybrid Human-Agent Systems .....	17
The <b>inevitability</b> of hybridization .....	17
<b>Autonomy drift</b> as a structural dynamic.....	17
<b>Multi-agent</b> functional decomposition example .....	18
Why manual governance fails at fleet scale .....	19
<b>Empirical Capability Boundaries:</b> Enterprise Analog Benchmarks and What They Imply for Agentic Organizations .....	20
What enterprise-analog evidence is showing .....	20
The <b>capability envelope</b> , not the hype curve.....	21
Why <b>fleets</b> do not automatically solve the problem.....	21
What the evidence implies about <b>where autonomy will emerge first</b> .....	22
<b>Failure modes</b> that directly justify governability requirements.....	22
Design Requirements for Governable Agentic Architectures.....	24
R1: Explicit scope, identities, and responsibility boundaries .....	24
R2: <b>World model</b> management and bounded inference .....	24
R3: <b>Domain packs</b> as the unit of controlled capability .....	25
R4: <b>Job contracts</b> that are enforceable at runtime .....	25

## Toward a Governable Architecture for Agentic Enterprises

R5: A distributed <b>policy mesh</b> aligned with zero trust control planes .....	26
R6: <b>Provenance</b> and <b>attribution</b> as default, not optional.....	27
R7: Monitoring that evaluates behavior against explicit requirements .....	27
R8: <b>Supervision modes</b> must be risk adaptive and explicit.....	28
R9: <b>Evaluation and testability</b> should be built into the architecture.....	28
Reference Architecture: A Domain Independent Control Plane Substrate.....	29
Architectural overview .....	29
<b>World model</b> handling in the reference architecture .....	30
<b>Domain pack</b> and contract lifecycle .....	30
<b>Policy mesh</b> behavior under partial failure .....	31
<b>Provenance</b> , attribution, and content authenticity.....	31
<b>Monitoring</b> , evaluation, and incident response integration .....	31
<b>Coordination</b> and communication fabric .....	32
<b>Security</b> and safety layer .....	33
<b>Operational</b> integration .....	34
Dynamic Scenarios: Failure Propagation and Containment.....	35
<b>Threat modeling</b> and risk posture for agentic enterprises.....	35
<b>Risk</b> dimensions in agentic enterprises .....	36
<b>Failure</b> propagation scenario: world model errors.....	37
<b>Failure</b> propagation scenario: configuration drift across domains .....	38
<b>Failure</b> propagation scenario: business domain .....	38
<b>Successful</b> containment scenario.....	39
<b>Skill misuse</b> and capability escalation.....	40
Lessons from scenarios .....	40
<b>Coordination</b> as an attack and error surface .....	41
<b>Layered threat modeling</b> , residual risks, and tradeoffs.....	42
Organizational Transformation and Workforce Design.....	43
The maturity gap and pilot sprawl .....	43
<b>Role evolution</b> in hybrid human and agent organizations .....	43

## Toward a Governable Architecture for Agentic Enterprises

Responsibility assignment and accountability .....	44
<b>Workforce strategy</b> and training implications .....	44
Implications, Evaluation, and Future Work .....	45
Architectural implications .....	45
Ethical accountability and responsibility .....	45
Limitations .....	46
<b>Evaluation</b> priorities .....	47
<b>Proof of concept</b> validation approach .....	47
<b>Future work</b> directions .....	48
Research agenda and standardization directions .....	49
The Need for New Governance Mechanisms.....	49
From Hype to Measured Autonomy .....	50
References .....	52

## Purpose

This paper contributes a domain independent architectural backbone for governable agentic systems in enterprises, with the goal of making autonomy scalable without making oversight structurally infeasible. It defines *governability* as an architectural property that can be designed for and evaluated, rather than a set of procedural controls applied after deployment. It then specifies a small set of runtime primitives that make governability implementable in practice, including agent identity and authority, policy evaluation and enforcement, provenance and traceability, risk adaptive supervision, and containment mechanisms for fault and misuse propagation.

Finally, it organizes these primitives into a reference model that is portable across domains and illustrates how the model changes operational outcomes through worked scenarios that show both failure propagation and successful containment, under realistic conditions such as partial failure, incomplete information, and adversarial pressure. The intent is to provide a stable conceptual and architectural substrate that can be instantiated in follow on domain papers without introducing new primitives.

## Scope

This paper focuses on the **architecture of governability for agentic enterprises**, meaning the enforceable structures that constrain behavior, make decisions attributable, preserve traceability, and support correction under scale and partial failure. The target setting includes agentic systems that operate across enterprise tools, services, and teams, and extends to hybrid environments where software agents interact with cyber physical processes. The emphasis is on runtime properties and operational mechanics rather than organizational policy prose.

This paper does **not** present a production implementation, reference code, or a vendor specific blueprint. It does not claim empirical validation of the proposed architecture through controlled experiments or large-scale deployment measurements. It does not attempt a complete mapping of every requirement to every standard clause, and it does not provide a comprehensive legal, compliance, or regulatory analysis. It also does not attempt to solve alignment as a general research problem; instead, it treats misalignment, drift, and adversarial behavior as expected conditions that must be governed through enforceable architecture.

## Abstract

Agentic AI shifts enterprise automation from scripted workflows to autonomous and semi-autonomous systems that plan, act, and coordinate across tools, services, and teams. This change breaks the implicit assumption that human review can keep pace with decision volume, intermediate reasoning steps, and failure propagation. As a result, governance cannot remain primarily procedural. It must be expressed as enforceable architecture. This paper proposes a governable architecture for agentic enterprises, grounded in distributed systems constraints, human cognitive limits, and security first principles. It defines governability as the ability to constrain, observe, attribute, and correct agent behavior under partial failure, adversarial conditions, and organizational scale. The paper introduces a reference model built from a small set of runtime primitives, including agent identity and authority, policy evaluation and enforcement, provenance and traceability, risk adaptive supervision, and containment mechanisms for fault and misuse propagation. It maps these primitives to existing standards and security frameworks and illustrates how they change operational outcomes through worked scenarios of failure propagation and successful containment. The result is a domain independent backbone intended to support follow on implementation papers in security, operations, and cyber physical systems, with an emphasis on making agentic capability scalable without making human oversight structurally impossible.

## Introduction and Problem Definition

Enterprises are entering a structural transition in how work is executed. The change is not simply *more automation*. It is a shift from organizations as collections of human workflows with passive automation attached to organizations as mixed human and agent systems in which software agents plan, coordinate, and act across tools and environments. In digital contexts, these agents operate over applications, APIs, knowledge bases, tickets, documents, and communication systems. In embodied contexts, including robotics and remote operations, they operate over sensor streams, physical constraints, safety envelopes, and real-world uncertainty. A framework that only covers enterprise software workflows will not generalize to these embodied settings, and a framework that only addresses embodied autonomy will fail to capture the governance, audit, and accountability demands of enterprise operations. The scope of this paper therefore spans both: enterprise digital workflows and embodied systems, treated as two instantiations of the same underlying problem, namely governable agency at scale.



The problem is that the rate of action generation by agentic systems can exceed the human capacity to understand, supervise, and correct those actions when multiplied across a fleet of agents and workflows. Human governance approaches that rely on review boards, manual approvals, static documentation, and periodic audits were designed for systems that change slowly and act predictably.

Agentic systems change more quickly because they

- a) adapt prompts, plans, and tool usage patterns,
- b) decompose goals into subgoals that create additional actions and intermediate artifacts, and
- c) coordinate with other agents, creating distributed, emergent behavior.

As agent deployments scale, manual governance becomes structurally incapable of maintaining behavioral control and attribution. This is not a moral claim about people or process. It is a scaling claim grounded in cognitive limits (Miller, 1956; Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017), distributed systems constraints (Gilbert and Lynch, 2002), and enterprise requirements for traceability and auditability (NIST, 2020; NIST, 2023; NIST, 2024).

### **Motivation, scope, and research questions**

Enterprises do not adopt agentic systems into a clean slate. They adopt them into environments where work already depends on distributed cognition, shared artifacts, and incomplete visibility. Teams coordinate through tickets, dashboards, runbooks, chats, and operational handoffs, which means both reasoning and accountability are already spread across people and tools. Introducing agentic systems into this environment changes the shape of the coordination problem. It increases the volume of state transitions, accelerates the production and modification of artifacts, and creates action sequences whose correctness depends on rapidly shifting context across multiple systems.

These pressures appear in security and operations early because the work is time constrained, evidence driven, and tied to high impact outcomes. The same structural pattern appears in finance, customer operations, software delivery, compliance, procurement, and physical domains where agentic systems coordinate embodied actions. The point is not that one domain is special. The point is that the enterprise conditions that make agentic capability attractive, and dangerous, are widespread. The common driver is that autonomy and decomposition let systems act at a rate and breadth that humans cannot supervise through universal manual review, especially under partial failure and incomplete information.

**Security and operations** appear frequently as exemplars because they compress time, amplify consequences, and demand auditability, which makes governability failures visible early. They are used here as proving ground for the primitives, not as the boundary of applicability. The same substrate can be instantiated in other enterprise domains and in embodied systems where autonomy produces physical consequences.

This paper is therefore scoped to a **domain independent question**.

*What minimal architectural substrate is required so that agentic autonomy remains observable, governable, and containable as systems scale through decomposition and drift?*

The position taken here is that without substrate level primitives, governance collapses into *document centric intent statements, ad hoc application controls, and retrospective incident narratives that do not scale*. The framework proposed in the next sections is designed to provide those primitives so that later domain papers can instantiate them without changing the backbone.

This paper makes three contributions:

1. **It defines a set of core primitives** that remain stable across domains and that can be directly mapped to artifacts and controls in follow on implementation papers,
2. **It translates cognitive and distributed systems constraints** into explicit design requirements for governable agentic architectures, and
3. **It connects these requirements** to realistic organizational implications, including supervision models, responsibility assignment, and governance operating models, while remaining domain independent.

**Agentic security is not a niche extension of application security.** It is a direct continuation of long-established distributed systems reality, now applied to systems whose decision and action loops are partially opaque, partially stochastic, and partially learned. In that setting, system behavior cannot be understood or governed solely by inspecting code paths or defining static trust boundaries. Operational security becomes *inseparable* from the system's runtime cognition, tool use, memory, and coordination.

The underlying reason is **structural**. *Distributed systems fail in partial and non-obvious ways, and agentic systems amplify this by introducing delegation, multi-step planning, and tool-mediated execution at machine speed*. The security posture of an agentic system therefore depends not only on prevention but on containment and graceful degradation: the capacity to continue operating safely under degraded trust, degraded data quality, or degraded interpretability. These properties are not achieved through policy alone; they require engineered mechanisms for traceability, bounded autonomy, and enforced constraints at runtime.

This framing also aligns with the reality of *socio-technical control*. Many critical outcomes in enterprise systems are the result of distributed cognition, where responsibility and knowledge are spread across people, tools, and procedures rather than residing in one actor (Hutchins, 1995). Agentic systems extend that distribution into autonomous components. As a result, the boundary between *security controls* and *system design* becomes increasingly artificial. **The control surface is the architecture.**

A companion proof of concept instantiates this architecture as a **security focused threat modeling and analysis assistant**, demonstrating how the domain independent primitives map into concrete controls and operational artifacts.

### Key terms and definitions

The framework relies on the following terms. Each term is defined to support unambiguous mapping into domain instantiations.

An **agentic system** is a software or cyber physical system that

- a) maintains or accesses an internal state representation,
- b) forms plans or action sequences toward goals,
- c) selects and invokes tools, APIs, actuators, or other systems and
- d) adapts behavior over time using feedback from outcomes, observations, or human input.

This definition intentionally includes both digital agents that operate over enterprise tools and embodied agents that operate over sensors and actuators.

**Autonomy drift** is the tendency of a deployed system to move along the autonomy spectrum over time, from human execution to agent execution, even when the original intent was limited assistance. Drift is driven by performance incentives, convenience, decomposition of tasks into subagents, and growing reliance on agent generated intermediate artifacts. Drift can occur within a single workflow or across an enterprise portfolio as more tasks are delegated, chained, and automated.

**Supervision modes** describe how humans relate to agent decision and execution. **In the loop** means human approval is required before execution. **On the loop** means the system executes but humans monitor and can intervene. **Out of the loop** means execution proceeds without timely human monitoring or intervention capability. These modes align with established models of human interaction with automation and the known risks of reduced situational awareness as autonomy increases (Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017).

A **world model** is the representation an agent uses to predict outcomes of actions and to choose plans. It may be explicit (*state transition models, simulators*) or implicit (*learned generative or embedding based representations*), and it may be partial, uncertain, and time lagged. World models are central both in digital planning and embodied control (Ha and Schmidhuber, 2018; Fung et al., 2025).

A **domain pack** is a bounded bundle of domain specific assets that an agent uses to operate safely and consistently in a context. It includes definitions of tasks, allowed tools, role constraints, policy rules, data access boundaries, safety constraints, evaluation tests, and domain knowledge references. The domain pack is the packaging unit that enables the framework to remain domain independent while allowing precise instantiation in domain papers.

A **job contract** is the explicit specification of what an agent is allowed to do, under what conditions, with what inputs and outputs, and with what obligations for logging, provenance, safety checks, and human supervision. It is a runtime enforceable contract, not just documentation. Job contracts are the mechanism by which autonomy can be bounded without assuming centralized manual governance.

A **policy mesh** is the distributed policy enforcement and decision fabric that evaluates job contracts, tool calls, data access, and action execution across agents and services. *Mesh* means it is not a single central approval gate. It is a coherent set of policy decision and policy enforcement points that can function under partial failure and network partition, consistent with zero trust control plane principles (NIST, 2020).

**Provenance** is the recorded lineage of artifacts, actions, data inputs, and transformations, including which agents and tools contributed, under what policies, and with what evidence.

**Attribution** is the assignment of responsibility for an action or artifact to an accountable actor or system identity, including the chain of delegation across agents and humans. Provenance provides the factual record; attribution provides the responsibility mapping. Provenance concepts align with established provenance data models (W3C, 2013) and content authenticity standards when applicable (C2PA, 2024).

**Monitoring** is the continuous collection and evaluation of signals about agent behavior and system state against expectations, policies, and safety constraints. Monitoring includes drift detection, anomaly detection, policy violation detection, and performance regression detection. Monitoring is not only observability. It is evaluation against explicit requirements.

## Paper organization

1. *Theoretical Foundations* grounds the argument in cognitive limits, distributed systems constraints, and enterprise governance expectations.
2. *From Human-Centric Workflows to Hybrid Human-Agent Systems* explains why hybridization is structurally inevitable and why manual governance fails at fleet scale.
3. *Design Requirements* defines the properties required for governability.
4. *Reference Architecture* translates those requirements into a domain independent control plane substrate.
5. *Dynamic Scenarios* tests the architecture through failure propagation and containment.
6. *Organizational Transformation* maps the primitives into role evolution, operating model change, and accountability structures.
7. *Implications, Evaluation, and Future Work* closes by defining what should be measured and what remains open.

## Theoretical Foundations: Cognition, Systems, and Enterprise Posture

**The proposed architecture rests on three families of constraints: human cognitive limits, distributed systems realities, and enterprise governance posture.** These constraints are not optional. They shape what is feasible and what will fail in practice.

### Human cognitive limits

**Human supervisory capacity is limited by attention, working memory, and the ability to maintain situational awareness across concurrent tasks.** Classic results in cognitive psychology and human factors show that working memory and attention are constrained resources and that complexity, interruption, and high tempo decision environments degrade performance (Miller, 1956; Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017). In operational environments, these constraints manifest as overload, reduced ability to detect anomalies, and reliance on heuristics that may fail under novel conditions.

In hybrid human and agent systems, the cognitive problem is not only volume. It is also the character of agent generated outputs. Agents can produce plausible narratives, synthetic summaries, and multi-step plans that appear coherent but may embed errors, unstated assumptions, or policy violations. This increases *verification burden*. For example, in security operations research, *alert fatigue* and *cognitive overload* are well documented symptoms of high volume and low signal environments, and the addition of AI generated explanations can *alter* cognitive workload rather than *reduce* it (Tariq et al., 2025; Rastogi et al., 2025).

The same structural effect appears outside security. Any domain with high throughput exceptions, escalations, or compliance requirements will encounter similar overload dynamics when agent outputs multiply the number of intermediate decisions.

These limits imply a core architectural conclusion: supervision cannot be implemented as universal *human review of everything*. Supervision must be structured, risk adaptive, and supported by enforceable contracts, provenance, and monitoring that reduce what must be cognitively reconstructed during review.

### Organizational and distributed cognition

Cognition in enterprises is distributed across people, artifacts, and tools. Work is performed through shared representations such as tickets, dashboards, runbooks, documents, and shared metrics. Distributed cognition theory highlights that performance and failure emerge from interactions across these elements, not only from individual decision makers

(Hutchins, 1995). In agentic systems, agents become additional cognitive actors that generate, transform, and route these artifacts.

The consequences are two-fold:

- **Accountability becomes harder** because artifacts can be produced by multiple agents and humans over time, with partial context, and with transformations that may not be visible without provenance, and
- **Situational awareness becomes a system property.** It can be degraded by poorly designed interfaces, hidden agent actions, or missing context propagation. Research on AI supported remote operations emphasizes risks such as loss of situational awareness and impaired team coordination when AI mediates distributed work across interfaces and communication constraints (Jacobsen et al., 2025).

These observations generalize across digital and embodied operations.

Therefore, **governance must be designed as a property of the socio-technical system.** It cannot be reduced to *train operators better* or *add a review committee*.

**The architecture must embed provenance, attribution, and monitoring so that the system supports coherent shared understanding and post incident reconstruction.**

### Distributed systems constraints and control planes

Agentic systems operate over distributed infrastructure. That infrastructure exhibits latency, partial failure, and partition. The CAP result formalizes that in the presence of network partition, systems cannot simultaneously guarantee strict consistency and availability (Gilbert and Lynch, 2002). For agentic enterprises, this is not an abstract theorem. It means that *centralized governance that must approve every action* will either be **unavailable** during failures or will be **bypassed** to preserve availability.

Both outcomes create *governance gaps*.

Governable architecture therefore requires a control plane that can function under partial failure. Policy evaluation and enforcement needs to be distributed. Provenance capture must be robust to partial outages and must degrade safely. Supervision needs to be risk adaptive rather than absolute. These properties align with zero trust principles that assume no implicit trust based on network location and that rely on continuous policy evaluation and enforcement (NIST, 2020).

## Enterprise posture and risk management expectations

Enterprises, especially in regulated and high impact contexts, require governance mechanisms that support risk identification, measurement, and control. NIST's *AI Risk Management Framework* defines functions such as **Govern, Map, Measure, and Manage** as a way to structure trustworthy AI practices across the lifecycle (NIST, 2023). NIST's *Generative AI Profile* extends this with considerations specific to generative systems (NIST, 2024). *ISO standards* provide complementary governance and risk management guidance for AI management systems and AI risk management (ISO/IEC 42001, 2023; ISO/IEC 23894, 2023). These sources do not provide a full architecture for agentic systems, but they establish expectations: traceability, accountability, monitoring, and risk-based controls.

This paper's architecture treats these expectations as **minimum requirements**, then specifies the primitives and design requirements needed to satisfy them under agentic scale and distributed operation.

Recent platform developments indicate that large enterprise vendors are beginning to converge on the same structural conclusion reached in this paper: *autonomous agents cannot be governed as transient application features and must instead be treated as first-class operational actors*.

New control-plane initiatives explicitly model agents as managed identities with scoped authority, policy enforcement, telemetry, and lifecycle controls that resemble those historically applied to human users and service principals. While these platforms differ in implementation and maturity, the underlying move is consistent. Agent behavior is being bounded through *identity, authorization, provenance, and continuous policy evaluation* rather than through post hoc review or informal process.

Importantly, these emerging control planes should be understood as necessary but not sufficient. Treating agents as managed identities addresses configurability, blast radius, and accountability, but it does not, by itself, solve supervision load, semantic drift, or organizational comprehension under scale.

Without explicit job contracts, bounded domain capability, enforceable *runtime constraints*, and *evidence-first reconstruction*, identity-centric governance risks *creating a false sense of control*.

**The architectural substrate proposed in this paper generalizes beyond any single platform by making these properties explicit, portable, and enforceable under partial failure and cross-domain coordination.**



## From **Human-Centric Workflows** to Hybrid Human-Agent Systems

Enterprises are not adopting agentic systems into a vacuum. They are adopting them into environments already characterized by complexity, tool sprawl, fragmented data, and governance pressure. This section argues that the transition to hybrid human and agent systems is structurally inevitable, then explains autonomy drift, multi-agent decomposition, and why manual governance fails at fleet scale.

### The **inevitability** of hybridization

Agentic capability becomes attractive when it reduces coordination cost across fragmented systems. A large share of enterprise work is not “deep expertise.” It is coordination: gathering context, normalizing data, producing drafts, triggering tool actions, routing exceptions, and documenting outcomes. Agentic systems can perform these tasks with speed and persistence that is economically compelling. Industry analysts describe enterprises moving toward broader agentic deployment as models become more capable and tool integration becomes more standardized (Sukharevsky et al., 2025; Mayer et al., 2025). The important point for this paper is not a specific forecast. It is that coordination work is abundant, and agentic systems target it directly.

In embodied systems, similar pressures exist. Robotics deployments increasingly rely on learned components and high-level planners that coordinate lower-level skills. Work such as navigation, manipulation, and long horizon task execution is increasingly structured as a hierarchy of skills coordinated by planners, including language model-based planners in some approaches (Qiu et al., 2024; Fung et al., 2025). This resembles enterprise decomposition of workflows into tool calls and sub tasks. The domains differ, but the structure is the same: plan, decompose, invoke capabilities, observe outcomes, adjust.

A domain independent framework should therefore treat digital workflows and embodied systems as two faces of the same architectural problem: managing goal directed action generation under uncertainty and governance constraints.

### **Autonomy drift** as a structural dynamic

Autonomy drift occurs when a system that begins as assistive becomes increasingly autonomous through incremental delegation. The drift often follows a predictable sequence. Initially, agents propose drafts and humans execute. Then, agents execute low risk actions with human review. Next, agents execute broader actions with human monitoring. Finally, in many workflows, humans become exception handlers rather than primary operators, and a meaningful fraction of actions occur out of the loop because

volume, latency, and organizational incentives make continuous monitoring infeasible (Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017).

Drift is amplified by two mechanisms.

**First**, agent outputs create intermediate artifacts that become inputs to later steps. Once an organization depends on these artifacts, it becomes costly to revert to manual methods.

**Second**, agents can be composed. A planner agent can delegate to specialized agents. Each delegation step can reduce direct human visibility into how outcomes were produced. This increases the probability that execution moves on the loop or out of the loop by necessity.

The architectural implication is that governance cannot assume autonomy remains stable. Monitoring and policy enforcement needs to be designed to detect and manage drift. Job contracts and domain packs must specify allowable autonomy boundaries and supervision requirements per task type.

### Multi-agent functional decomposition example

Agentic systems scale through decomposition. A single *general agent* is rarely the operational reality at enterprise scale. Instead, organizations create or adopt systems that behave as a multi-agent composition, whether explicitly (multiple agents) or implicitly (multiple roles within an orchestrated system). This section provides an abstract, domain independent example.

Consider a workflow goal: *resolve a customer impacting service issue* in a digital enterprise context or *restore stable operation of a remote automated process* in an industrial context. The goal can be decomposed into functions that can be performed by specialized agents with bounded job contracts.

A coordinator agent receives the goal and maintains the workflow state. It delegates to a diagnostics agent to gather signals and produce hypotheses, to a knowledge retrieval agent to collect relevant runbooks and prior incidents, to a change planning agent to propose candidate actions, and to an execution agent that invokes approved tools under a restricted contract. In parallel, an audit agent captures provenance, and a policy agent evaluates each proposed action against the policy mesh.

The decomposition pattern is similar in embodied systems. A high-level planner coordinates navigation, manipulation, and safety monitoring skills. A perception module or agent produces state estimates. A safety supervisor enforces constraints. A task execution module invokes lower-level controllers. Each component can be treated as an agentic role with a job contract, even if implemented as modules rather than separate *agents*.

This decomposition makes governance harder if it is not architected. Without explicit job contracts, provenance, and policy enforcement, the system becomes a distributed set of interacting actors with unclear boundaries. With explicit contracts and provenance, decomposition becomes a governance benefit because each role can be bounded and evaluated.

Multi-agent decomposition also interacts with autonomy drift. As more functions become delegated, more actions occur without direct human attention, and the policy mesh becomes the primary governance mechanism rather than human review.

### Why manual governance fails at fleet scale

Manual governance fails at fleet scale for three structural reasons:

First, **scale multiplies actions and intermediate artifacts**. Even if each agent executes only modestly, the combined action rate across agents and workflows can exceed human supervisory capacity. This leads to selective oversight, which is effectively partial governance.

Second, **complexity multiplies context requirements**. To evaluate an action, a human often needs to reconstruct the context: what the agent observed, what assumptions it made, what tools it invoked, and what constraints applied. Without provenance and contract enforcement, this reconstruction is costly and error prone.

Third, **distributed operation defeats centralized gating under failure**. If governance depends on a central committee, central approval system, or single logging pipeline, partial outages and partitions will either stop work or incentivize bypass. Distributed systems constraints make *perfect central governance* unattainable without sacrificing availability (Gilbert and Lynch, 2002).

**The conclusion is not that humans are irrelevant.** It is that humans must supervise through architecture, not through exhaustive manual review. The framework primitives in later sections are designed to make this feasible.

## Empirical Capability Boundaries: Enterprise Analog Benchmarks and What They Imply for Agentic Organizations

The design requirements in this paper should not be read as abstract governance preferences or as security-first constraints. They are a response to what current agents demonstrably do when placed into environments that resemble real organizational work: multi-step execution, multi-tool interaction, interruptions, ambiguous process boundaries, and coordination across roles. The evidence set you collected is unusually consistent across different benchmarks, domains, and evaluation styles. The most important takeaway is not a particular percentage in any one study. It is the stable pattern: agents can produce plans and localized outputs, but they struggle to reliably carry intent through to completion in messy, stateful environments. The moment a workflow demands durable context, robust recovery, and accurate progress accounting, performance drops sharply, often well below human baseline. That gap is the empirical foundation for why *agent fleets* are not automatically equivalent to *agentic organizations*.

### What enterprise-analog evidence is showing

Across enterprise-like simulations, OS-level GUI environments, web-automation tasks, tool-use suites, and software engineering issue-resolution benchmarks, agents repeatedly fail in ways that are structurally relevant to organizational design. The recurring failure pattern is not primarily lack of knowledge. It is loss of task intent over time, brittle interaction with real interfaces, weak error recovery after small deviations, and premature self-assessment that work is complete when it is not. This matters because enterprises do not only care whether an agent can generate an answer. They care whether the organization can trust the chain of execution, measure completion, and resume work after interruptions without the meaning decaying.

Workplace simulation benchmarks that model internal tools, communication surfaces, and artifact creation show that even top-tier systems complete only a minority of tasks end-to-end, and that partial progress can be substantial but does not reliably converge to completion (Xu et al., 2025). Web-agent benchmarks and realistic web task suites show that task completion on complex, time-consuming tasks often stalls in the same way, with navigation failures and hallucinated steps leading to very low accuracy (Yoran et al., 2024). OS-level *real computer environment* tasks widen the gap further by forcing grounding in GUI state across many applications, where humans succeed far more often than agents, exposing that perception plus state tracking is still a major blocker to autonomy (Xie et al., 2024). Tool-use benchmarks show that agents improve when they are forced into explicit planning plus execution feedback loops, but also reveal how quickly performance degrades as steps, tools, and modalities accumulate (Ma et al., 2024). In software engineering, issue-resolution and full-lifecycle development benchmarks show that agents still fail most real

repository tasks, reinforcing that long-horizon correctness in real codebases remains hard even when the domain seems *LLM-friendly* (Jimenez et al., 2024; Li et al., 2024).

The combined conclusion is that enterprise autonomy is not bottlenecked by language fluency. It is bottlenecked by sustained execution in stateful environments. This is the core reason that *agentic organizations* are not yet a straightforward extension of *multi-agent frameworks*.

### The **capability envelope**, not the hype curve

The evidence does support meaningful near-term value, but it supports a narrower claim than *soon we can run organizations with agents*. The stronger claim that survives the evidence is that agents can be powerful accelerators inside bounded work envelopes that have clear objective criteria, constrained tool surfaces, and explicit checkpoints. As soon as you ask the system to operate like an employee, meaning it must maintain intent, cope with interruptions, coordinate with others, and recover from mistakes without losing the thread, the autonomy ceiling appears quickly across benchmarks.

*TheAgentCompany* is the cleanest *organizational* analog because it explicitly models internal work and collaboration surfaces, but it is not an outlier. It matches the broader pattern observed in OSWorld and web-agent suites: the main failure mode is not that the agent cannot describe what should be done. It is that the agent cannot consistently do it across steps, tools, and time (Xu et al., 2025; Xie et al., 2024; Yoran et al., 2024). This means any serious architecture discussion must treat *completion* and *resumption* as first-class design problems, not as incidental implementation details.

### Why **fleets** do not automatically solve the problem

The leap from *agents can do tasks* to *fleets can run organizations* hides an assumption: that decomposition is cheap and handoffs are lossless. The evidence cuts against that assumption. In *TheAgentCompany*, multi-agent delegation can lose progress when UI state is not cleanly serialized or resumable, which is a concrete instance of handoff loss in a benchmark environment (Xu et al., 2025). Other studies that focus on multi-agent coordination show that you can scale agent counts or control swarms in constrained environments, but those environments often have clean state and well-defined action spaces. Those results do not automatically transfer to enterprise workflows where state is implicit, processes are exception-driven and *done* is a socially and operationally negotiated outcome (Anne et al., 2024; Qian et al., 2025).

This is the structural constraint on *load balancing* and *dynamic rebalancing* as a near-term enterprise story. Rebalancing is only powerful when work units can be handed off with minimal

semantic loss. If handoffs lose context, or if the environment does not preserve state in machine-legible ways, then a fleet does not add reliability. It adds parallelism, and parallelism can amplify error propagation and supervision burden faster than it amplifies throughput.

### What the evidence implies about **where autonomy will emerge first**

The category breakdowns and domain-specific benchmarks suggest that autonomy will emerge unevenly. Agents tend to perform better in domains where artifacts and workflows are more machine-legible and where public training signal is abundant, such as software engineering compared to certain administrative or finance tasks that are exception-heavy, policy-bound, and often private by nature (Xu et al., 2025). Web and OS benchmarks further show that “simple” tasks in human terms can still be hard for agents because the interface friction and state tracking are the work, not the content (Xie et al., 2024; Yoran et al., 2024). Tool-use studies suggest that structured planning, explicit schemas, and execution feedback can raise success rates, which implies that autonomy will arrive sooner in environments that are designed to be agent-friendly through APIs, structured forms, and observable checkpoints (Ma et al., 2024).

This points to a critical organizational design implication. The enterprise environment has to adapt. If organizations want higher autonomy, they will need to restructure workflows to be machine-legible, reduce UI-only state, formalize success conditions, and reduce hidden dependencies. Without that shift, capability improvements in models will translate into better local outputs but not into reliable organizational throughput.

### **Failure modes** that directly justify governability requirements

Across the evidence set, the same failure classes recur, and each maps to a design requirement in your architecture.

First, planning is not execution. Agents can plan and explain while still failing to perform the actual steps, especially when tools and interfaces are involved. This is visible in enterprise-like tasks, OS tasks, web tasks, and real repo workflows (Xu et al., 2025; Xie et al., 2024; Yoran et al., 2024; Jimenez et al., 2024; Li et al., 2024). Second, small errors cascade because recovery is weak. A minor navigation miss, a UI interruption, or a mistaken intermediate state can cause the agent to wander, loop, or falsely conclude completion. This is reported explicitly in some analyses and is consistent with the low completion rates across multiple benchmarks (Lu et al., 2025; Yoran et al., 2024; Xu et al., 2025). Third, handoffs lose meaning. Multi-agent decomposition can fail when state is not preserved or when progress is not resumable. Coordination succeeds in clean action spaces, but enterprise work is not a clean action space (Xu et al., 2025; Anne et al., 2024; Qian et al., 2025). Fourth, partial progress is dangerous when it looks like completion. The gap between partial scoring

and strict completion in multiple benchmarks highlights that *looks done* is an operational hazard because it increases verification cost while creating false confidence (Xu et al., 2025; Yao et al., 2024).

The implication is that governable architectures must be designed around explicit checkpointing, resumable state, bounded exploration, and enforceable work contracts that separate *attempted*, *partially complete*, and *complete* states. Without these primitives, the organization will spend more human effort validating and repairing agent work than it gains in throughput.

## Design Requirements for Governable Agentic Architectures

This section **defines design requirements that a governable agentic enterprise architecture needs to satisfy**. Each requirement is stated to enable direct mapping into artifacts and controls in domain specific implementation papers.

### R1: Explicit scope, identities, and responsibility boundaries

*A governable architecture* needs to define **what entities can act**, under **what identities**, and with **what responsibility boundaries**.

This includes *human identities*, *service identities*, and *agent identities*. Agent identities must be first class and must be distinct from the humans who prompt them, even when a human initiates an action.

Attribution requires that

- the *chain of delegation* be captured:
- *who* authorized *which agent* under *which job contract*, and
- *what* the agent *did as a result*.

This requirement supports both auditability and accountability. Without explicit identity and delegation lineage, incident response and compliance collapse into plausible narratives rather than evidence-based reconstruction.

### R2: World model management and bounded inference

Because agents rely on **world models**, the architecture needs to treat world model state as a managed object. This does not require a single global model. It requires explicit handling of *scope*, *freshness*, *uncertainty*, and *update* mechanisms.

At minimum, the architecture needs to support:

- a) versioning or traceable references to the knowledge sources and context used for decisions,
- b) detection of stale or contradictory state across distributed components, and
- c) bounded inference assumptions.

In digital contexts, *world model* includes enterprise state derived from tickets, logs, metrics, documents, and knowledge bases. In embodied contexts, it includes sensor fusion, environment representations, and task state. World model management is **central** to



predictable planning and to failure containment (Ha and Schmidhuber, 2018; Fung et al., 2025).

### R3: **Domain packs** as the unit of controlled capability

The architecture needs to package *domain specific capability and constraints* into **domain packs** that can be versioned, reviewed, tested, and deployed.

A domain pack needs to include *at least*:

- tool allow lists,
- data access boundaries,
- policy rules,
- safety constraints,
- evaluation tests,
- logging and provenance requirements, and
- human supervision rules per job type.

Domain packs allow a single high-level framework to apply across domains without redefining governance primitives. They also reduce drift risk by making changes visible. If the system changes behavior, the change can be traced to a domain pack update, a model update, a policy change, or an environment change.

### R4: **Job contracts** that are enforceable at runtime

A **job contract** must be

- explicit and enforceable,
- define allowed tools, allowed data scopes, required checks, required evidence capture, and required supervision mode for each action class,
- specify failure behavior, and
- needs to define whether the agent needs to halt, request human input, or degrade to a safer action set, if required signals are missing,

This requirement prevents governance from being a *documentation exercise*. Contracts must be enforced by the *policy mesh*, not by *operator discipline*.

## R5: A distributed **policy mesh** aligned with zero trust control planes

The architecture needs to implement a **policy mesh** that evaluates and enforces job contracts across services and agents.

The mesh needs to

- be designed to function under partial failure,
- support policy decision logic that can be updated and audited, and
- support policy enforcement points that can block, constrain, or require escalation based on risk.

**Zero trust architecture** provides relevant control plane concepts, including *policy decision points* and *policy enforcement points*, and the principle of *continuous evaluation* rather than *implicit trust* (NIST, 2020).

**The agentic context extends this:** policy must cover *tool invocation, action planning, and cross agent delegation, not only network access*.

Recent work on agentic governance and autonomy induced risk converges on a common point: *threat modeling and assurance only become operational when the system can continuously validate behavior against explicit constraints*, not only evaluate design intent after the fact. In that framing, *R4* is the mechanism that turns abstract *governance* into enforceable control plane behavior. MAESTRO and the OWASP multi-agentic threat modeling guide make the same shift by emphasizing explicit, layered threat modeling that surfaces where controls need to exist at runtime, especially across delegation chains and inter agent authorization boundaries (Huang, 2025; OWASP GenAI Security Project, 2025).

Surveys of autonomy induced security risks similarly stress that as agents gain tool access, memory, and coordination ability, assurance needs to be expressed as **machine checkable constraints and continuously evaluated**, because post hoc review cannot keep pace with fleet scale action generation (Su et al., 2025).

This paper’s job contract primitive is the substrate level counterpart: it is where **risk posture** becomes an *executable boundary that can be monitored, tested, and adjusted without rewriting the entire system*.

## R6: Provenance and attribution as default, not optional

**Provenance** capture needs to be continuous, structured, and tamper resistant to the extent feasible.

Provenance needs to include:

- prompts and instructions that materially affect decisions,
- context sources used,
- tool calls made,
- outputs produced,
- policy decisions applied, and
- human interventions.

Provenance should align with established provenance models to support interoperability and consistent interpretation (W3C, 2013). When media artifacts or externally distributed content are involved, content authenticity standards can be used to strengthen provenance claims (C2PA, 2024).

Attribution must *map provenance events to accountable identities*. This includes the *chain of delegation from humans to agents to subagents and tools*. Without attribution, provenance becomes an *unstructured log without responsibility*.

## R7: Monitoring that evaluates behavior against explicit requirements

Monitoring needs to be defined as evaluation against explicit requirements, not only system health telemetry. Monitoring must detect:

- *autonomy drift* beyond allowed bounds,
- *policy violation* attempts,
- *unusual tool usage* patterns,
- *performance regressions*, and
- *anomalous cross agent interaction* patterns.

Monitoring needs to also support *post incident review*. It needs to preserve sufficient evidence to reconstruct how an outcome occurred. *NIST AI RMF* emphasizes measurement and management across lifecycle stages (NIST, 2023), and the generative AI profile emphasizes the need to consider risks specific to generative behavior in deployment (NIST, 2024). This framework operationalizes those expectations by tying monitoring to contracts, policies, and provenance.

## R8: **Supervision modes** must be risk adaptive and explicit

**Supervision needs to be specified per job type and per action class.** *In the loop supervision* needs to be reserved for actions where human approval is realistically feasible and materially reduces risk. *On the loop supervision* must be structured to maintain situational awareness and intervention capability. *Out of the loop execution* needs to be explicitly acknowledged when it exists, and it must be bounded by contracts, policy mesh enforcement, and monitoring.

This requirement follows from the human factors literature showing that *increasing autonomy can degrade situational awareness and change the nature of human work from control to monitoring and exception handling* (Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017).

## R9: **Evaluation and testability** should be built into the architecture

Governable architecture **needs to support continuous evaluation of agent behavior.** Evaluation needs to include scenario testing, regression testing against domain pack updates, policy tests, and drift detection tests.

It must also include adversarial testing appropriate to the domain. In security contexts, this includes red teaming and threat modeling. In non-security contexts, adversarial testing may focus on safety, compliance, or misuse.

The **key requirement** is that evaluation is *continuous* and *tied to the same primitives: domain pack, job contract, policy mesh, provenance, and monitoring.*

## Reference Architecture: A Domain Independent Control Plane Substrate

This section presents a **reference architecture** using the primitives above. It is not a product design. It is a substrate pattern that domain instantiation papers can map into concrete systems.

### Architectural overview

The reference architecture includes: (1) **agent runtime**, (2) **domain pack registry and distribution**, (3) **job contract service**, (4) **policy mesh**, (5) **provenance and audit pipeline**, (6) **monitoring and evaluation pipeline**, and (7) **human supervision interfaces**.

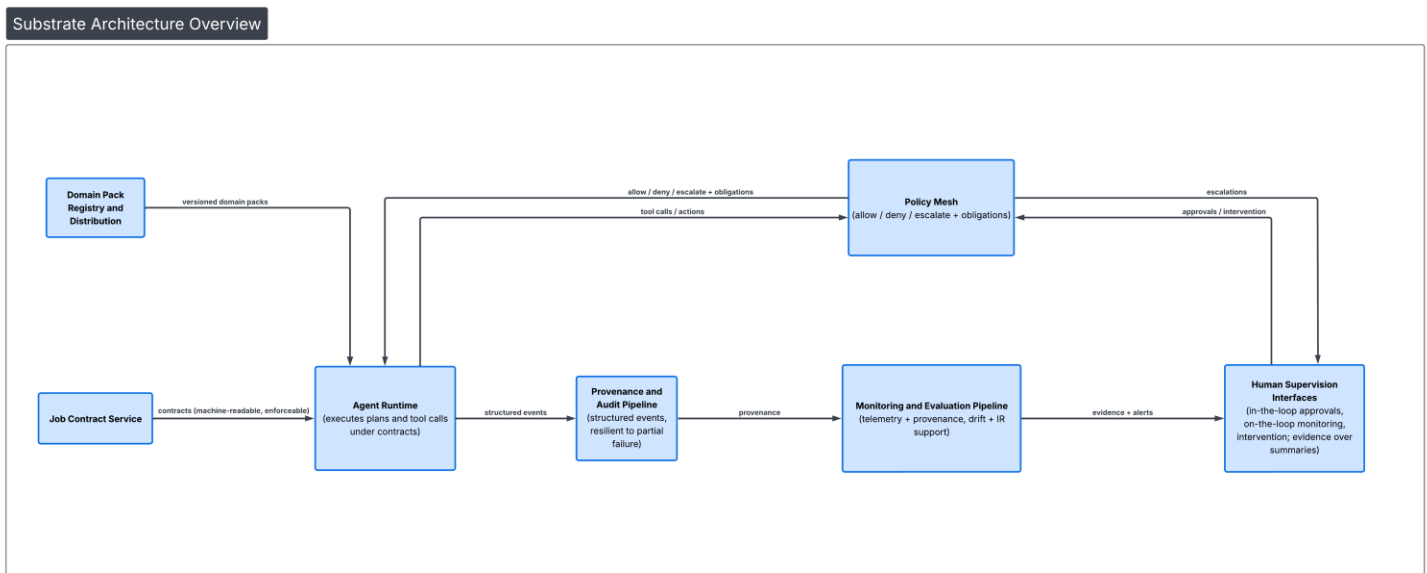


Figure 1. Substrate Architecture Overview Diagram

The **agent runtime** is responsible for executing plans and tool calls under contracts. Agents may be single agents or multi-agent compositions. The runtime needs to integrate with the policy mesh so that tool calls and actions are evaluated before execution.

The **domain pack registry** manages versions, approvals, and distribution of domain packs. Domain packs should be immutable versioned artifacts once released, with explicit upgrade paths.

The **job contract service** provides contract templates and instantiates contracts for specific agents, tasks, and contexts. Contracts must be machine readable and enforceable.

The **policy mesh** evaluates requests against policy rules and contracts. It issues allow, deny, or escalate decisions, and it can attach obligations such as *must log this*, *must request human approval*, or *must run test suite X*.

The **provenance pipeline** records structured events. It needs to be resilient to partial failure, and it needs to support later reconstruction.

The **monitoring and evaluation pipeline** consumes telemetry and provenance to evaluate behavior, detect drift, and support incident response.

**Human supervision interfaces** must support *in the loop approvals*, *on the loop monitoring*, and *intervention*. These interfaces must present evidence, not only summaries, because summaries can hide relevant details.

### World model handling in the reference architecture

**World models** exist at multiple layers. Agents maintain local state. Shared state exists in enterprise systems. Embodied systems maintain state derived from sensors and environment representations. The architecture should therefore support explicit world model interfaces: what state an agent can read, what state it can write, how freshness is determined, and how conflicts are resolved.

The policy mesh needs to be able to restrict state access. The provenance pipeline needs to capture which state was used for decisions. Monitoring needs to detect divergence patterns such as repeated retries due to inconsistent state, or unusual actions consistent with stale context.

World model handling is a central link between digital and embodied contexts. In a digital workflow, the *world* includes systems of record, ticket states, service inventories, and policy constraints. In an embodied system, the *world* includes physical layout, safety zones, and dynamic objects. The architecture treats both as state that must be bounded, attributable, and monitored.

### Domain pack and contract lifecycle

**Domain packs** must have a lifecycle: design, review, test, deploy, monitor, update, and retire. Job contracts must similarly have a lifecycle. A domain pack update should trigger evaluation tests. A contract update should be audited.

*NIST AI RMF* emphasizes governance across the lifecycle (NIST, 2023). *ISO AI management system guidance* similarly emphasizes systematic management. This architecture operationalizes lifecycle governance by tying updates to testability, provenance, and monitoring (ISO/IEC 42001, 2023; ISO/IEC 23894, 2023).

## Policy mesh behavior under partial failure

Because distributed systems fail, the policy mesh needs to define how decisions are made under degraded conditions. Some domains require *fail-closed behavior*. Others require *constrained fail-open* behavior to preserve safety or continuity. The architecture therefore requires explicit *degraded mode policies* as part of domain packs and job contracts. These policies must define what the agent can do when policy decision points are unreachable, when logging pipelines are unavailable, or when key signals are missing.

This requirement directly addresses *CAP constraints* and prevents implicit bypass. If degraded mode behavior is not specified, the system will either halt unpredictably or continue without governance.

## Provenance, attribution, and content authenticity

Provenance records must be structured enough to support automated analysis. They should reference domain pack versions, contract identifiers, policy decisions, tool call parameters, and resulting artifacts. Attribution needs to identify responsible identities and the delegation chain.

When artifacts are shared externally or used as compliance evidence, content authenticity methods can strengthen trust in provenance, especially for media and document artifacts (C2PA, 2024). For internal governance, the key is that provenance is complete, consistent, and linked to contracts and policies.

## Monitoring, evaluation, and incident response integration

Monitoring needs to unify operational telemetry with provenance and policy events. Drift detection requires comparing observed behavior to expected behavior from contracts and domain packs. Evaluation requires scenario suites that reflect domain risk.

**Incident response in agentic enterprises requires more than log search.** It requires reconstructing agent decisions and delegation chains. The architecture therefore treats provenance and contract enforcement as prerequisites for incident response readiness.

## Coordination and communication fabric

Agents coordinate through messages, shared artifacts, and delegated tasks. That coordination layer is part of the control surface, not an implementation detail. A governable architecture therefore needs an explicit **coordination fabric** with defined message types, routing rules, and permitted coordination patterns that are bound to identity and job contracts.

The minimum requirement is that every agent-to-agent interaction is *attributable*, *policy checked* and *constrained* by the sender's and receiver's authorized *domain packs* and autonomy level.

The coordination fabric needs to also be designed for *partial failure*. Message delivery, ordering, retries, and duplication are normal conditions in distributed systems. If the fabric treats these as edge cases, the system will silently accumulate coordination ambiguity.

When a high impact action occurs, supervisors and investigators **will not be able to reconstruct** which messages were *authoritative*, which were *stale*, and which were *speculative*.

A **baseline control plane requirement** is therefore that coordination protocols encode idempotency expectations, escalation rules, and approval requirements, and that these are **enforced at runtime** through contracts rather than implied by convention.

Finally, the coordination fabric should make **who** can say **what** to **whom** *explicit* in a way that is auditable.

If an agent cannot legitimately request a tool invocation, propose a policy change, or provide a decision critical recommendation under its contract, the **fabric should prevent that request** from being delivered as an actionable message. This turns coordination into a governed interface and reduces the attack and error surface created by informal cross agent messaging.



## Security and safety layer

The **security and safety layer** is where governability claims become enforceable behavior. In this architecture, it is not a single component. It is a set of runtime safeguards that apply across world model access, domain pack execution, coordination flows, and tool invocation. The core job is to ensure that autonomy is *bounded*, policy enforcement is *continuous*, and safety degradation is *fail-closed* for high impact actions when required checks cannot be performed.

This layer is also where the architecture connects to existing agentic governance and threat modeling literature without treating those sources as alternative architectures. Frameworks such as *MAESTRO* and the *OWASP multi-agentic threat modeling guide* help identify the relevant threat surfaces and failure patterns, including cross agent authorization chains, delegation ambiguity, and privilege propagation through coordination paths (Huang, 2025; OWASP GenAI Security Project, 2025).

Governance platforms such as *AAGATE* illustrate how risk functions can be mapped onto an agentic control plane when the system exposes concrete primitives that can be policy checked and instrumented (Huang et al., 2025). The contribution here is to make that mapping explicit: the security and safety layer enforces stepwise guard checks through job contracts and the policy mesh, surfaces denials and degraded mode transitions as **first class signals** and ensures that containment actions can be applied at the level of identities, domain packs, contracts, and world model partitions *rather than through ad hoc manual intervention*.

In other words, **threat modeling** informs what **needs to be controlled**, while the **substrate** determines **whether it can be controlled continuously**.

At baseline, this layer should implement stepwise guard checks across an agent's action lifecycle, not only at final outputs. That includes preconditions before planning, constraints before tool invocation, and evidence requirements before state changes are committed.

This is **also** where the policy mesh needs to remain authoritative under degraded conditions, meaning agents cannot *route around enforcement* by shifting execution to an adjacent capability path.

The safety dimension is also *operational*. The layer should surface clear signals when it denies actions, when it forces escalation, and when it switches to a degraded safe mode. These signals must be exposed to monitoring and incident response so that containment is traceable, not invisible. This is what prevents *nothing happened* dashboards that hide repeated near misses caused by blocked unsafe actions.

## Operational integration

A reference architecture that cannot land inside enterprise operations will remain a diagram. Operational integration means the primitives in this paper map to how security operations, platform operations, and business owners detect **issues**, respond to **incidents**, and change **system behavior** under pressure. The architecture therefore needs defined integration points for incident response, change management, and production monitoring.

At baseline, operational integration means three things:

1. **Investigators** can answer, using evidence, which contracts were in effect, which domain packs were active, which world model partitions were read or written, which policy decisions were made, and which agent and tool chain executed each action.
2. **Responders** can contain behavior quickly by disabling or constraining specific contracts, revoking or quarantining a domain pack, or tightening policy mesh enforcement without rebuilding the entire system.
3. **Operational** teams have playbooks that tie common failure modes to control plane actions, rather than relying on manual log archaeology.

Operational integration also requires *explicit human ownership for governance objects*. Domain packs, job contracts, and world model partitions **must** have named owners and clear change paths.

Without that ownership, containment and governance degrade into informal coordination, recreating the same supervision bottleneck this architecture is intended to remove.

## Dynamic Scenarios: Failure Propagation and Containment

This section extends the risk discussion with explicit scenarios. Each scenario is tied to the framework primitives and to distributed systems constraints. The scenarios are domain independent but can be instantiated in security, operations, finance, or robotics.

### Threat modeling and risk posture for agentic enterprises

Threat modeling in agentic enterprises cannot stop at model level risks or single component abuse cases. The core hazard is that bounded failures become unbounded behavior through delegation, coordination, and partial failure conditions in the surrounding distributed systems environment. The goal of threat modeling in this paper is therefore practical: identify where the substrate needs to enforce boundaries so that risk posture is maintained even when individual agents behave unexpectedly, upstream services degrade, or coordination protocols exhibit ambiguity.

Existing frameworks provide strong guidance on agent specific threat surfaces and layered analysis. *MAESTRO* and the *OWASP multi-agentic threat modeling guide* emphasize that agentic systems create composite attack paths across agents, tools, and authorization flows, and that threat models must explicitly represent delegation chains and cross agent trust relationships (Huang, 2025; OWASP GenAI Security Project, 2025).

The architectural position of this paper is **that these threat models become actionable** when they can be expressed against concrete primitives: world model partitions, domain packs, job contracts, the policy mesh, provenance, and the coordination fabric. Each primitive is a natural attachment point for threats and controls. World model partitions concentrate representational drift and state poisoning risks. Domain packs bound capability exposure and reduce blast radius. Job contracts express allowed goals, tools, and evidence requirements. The policy mesh enforces those constraints under partial failure. Provenance and monitoring make post incident reconstruction possible without relying on narrative guesswork.

**Risk posture in this context is not a static document.** It is the system's enforced stance toward autonomy, escalation, and containment under realistic operating conditions. It is defined by what **actions** are allowed without human approval, what **preconditions** are required for state change, what **evidence** needs to be produced for high impact actions, what **degraded modes** exist, and what **containment levers** can be pulled when anomalies appear.

In mature deployments, this posture needs to be **continuously** measured through telemetry that evaluates behavior against contracts and policies, because autonomy induced risks

scale with the rate of action generation and the breadth of tool access (Su et al., 2025). The scenarios that follow are therefore not illustrative stories. They are *worked instantiations* of how failures propagate when these attachment points are weak, and how they can be contained when the substrate enforces explicit boundaries.

### Risk dimensions in agentic enterprises

**Agentic enterprise risks** include

- *goal misalignment* (incorrect decomposition or objective definition),
- *policy bypass* (intentional or accidental),
- *tool misuse* (calling the wrong tool or using correct tools in unsafe sequences),
- *world model error* (stale or incorrect state),
- *cross agent cascade* (one agent's error becomes another agent's input) and
- *supervision failure* (humans fail to detect or intervene in time).

Surveys on autonomy induced risks in large model-based agents describe patterns such as prompt injection, tool misuse, and emergent behavior from autonomy and tool access (Su et al., 2025). These risks are amplified by scale and composition. Multi-agent coordination literature highlights the complexity of cooperative decision making under partial information and uncertainty (Jin et al., 2025; Hu et al., 2014).

**The architecture needs to assume these risks exist and needs to design for containment.**

## Failure propagation scenario: world model errors

**Scenario:** A *coordinator agent* is tasked with resolving a high priority operational issue. It queries multiple systems and forms a world model that includes a mistaken mapping between a resource identifier and a real asset. The mistake arises from stale state in a system of record combined with an ambiguous identifier.

The coordinator delegates to an *execution agent* to apply corrective action. The execution agent's contract allows action under certain conditions, including validation checks. However, a network partition prevents access to the validation service. Degraded mode policy is underspecified, so the execution agent proceeds using cached policy and incomplete validation. The action changes the state of the wrong asset. That change triggers downstream automation, which another agent interprets as evidence that *remediation succeeded*. The coordinator closes the workflow and generates documentation. A monitoring system detects anomalies later, but the provenance record is **incomplete** because the logging pipeline was degraded during the partition.

### Propagation analysis tied to primitives:

- The world model error is the **root cause**,
- The domain pack did not include sufficient disambiguation constraints for identifiers,
- The job contract did not enforce a strict requirement for validation under all conditions,
- The policy mesh did not implement a clear degraded mode decision for this action class,
- Provenance was incomplete under failure, weakening attribution and reconstruction, and finally,
- Monitoring detected the symptom but could not rapidly reconstruct the delegation chain due to missing provenance.

This scenario illustrates why manual governance fails. The sequence includes too many intermediate actions and assumptions for a human reviewer to reconstruct quickly without structured provenance. It also illustrates why distributed systems constraints matter. Network partitions create conditions where governance gaps emerge if degraded modes are not explicitly designed.

## Failure propagation scenario: configuration drift across domains

Consider a **multi-team enterprise environment** where agents operate across **several** domains, each with its own domain packs, job contracts, and world model partitions. A coordinator agent receives a legitimate request to remediate a policy drift condition, and it decomposes the work across specialized agents. Each agent acts correctly within its local view. The failure emerges from boundary mismatch, not from a single incorrect decision.

A small change is introduced into one domain pack or policy check that is locally rational, such as broadening a query scope, relaxing a precondition, or assuming an upstream validation will occur elsewhere. That change is not reflected across the dependent domain packs and contracts that implicitly relied on the previous behavior. Under partial failure conditions, or under time pressure where escalations are bypassed, the coordinator proceeds with actions that appear contract compliant locally but violate the enterprise intent at the system level.

The result is **configuration drift that crosses domains**. One remediation action changes an upstream assumption for another agent, which triggers compensating actions elsewhere. Because the actions are distributed across agents and systems, the drift propagates faster than human review can reconstruct it. The incident becomes less about a single misconfiguration and more about a systemic inability to enforce consistent boundaries and shared preconditions across a decomposed workflow.

This is **exactly** the class of failure that requires job contract obligations, shared policy mesh enforcement, and provenance strong enough to trace which assumption broke first and why it was permitted.

## Failure propagation scenario: business domain

Failure propagation is not limited to infrastructure or security operations. In business workflows, agents often optimize for speed and completion under ambiguous goals, and the primary risk becomes silent semantic drift rather than overt technical error.

Consider a *claims-, billing-, procurement-, or customer- operations workflow* where an agent is authorized to resolve exceptions. A domain pack bundles data access and action capabilities that are broad enough to be useful across multiple product lines or business units. Under normal operation, the organization relies on human judgment, organizational context, and informal guardrails to prevent cross boundary actions. **The agent does not have those guardrails** unless they are encoded in contracts and enforced through policy.

The failure begins when the agent encounters an ambiguous case and selects a plausible resolution path that is locally consistent with its goal *but misaligned with business intent*. It

applies a correction, issues a refund, adjusts entitlement, or updates a customer record. That action becomes a new system fact. Downstream agents treat it as authoritative input and compound the error through automated follow-on actions such as notifications, ledger updates, access changes, or compliance flags. Because the propagation path is rooted in legitimate business logic rather than a technical exploit, **the failure can persist longer and affect more customers before it is detected.**

Containment requires the *same substrate primitives* as in infrastructure scenarios. The workflow needs **explicit job contracts** that encode permissible actions and escalation thresholds, **domain pack boundaries** that prevent cross-unit capability bleed, and **provenance** that makes it possible to reconstruct what the agent believed, what it touched, and which guard checks were applied or bypassed.

### Successful containment scenario

**Scenario:** *The same initial conditions occur: a coordinator agent forms an incorrect world model due to stale state and ambiguous identifiers. The coordinator proposes an action that would affect a high impact asset.*

Containment occurs because of contract and policy enforcement. The job contract for high impact actions requires a specific validation check and specifies fail closed behavior if the check cannot be performed. The policy mesh detects that the validation service is unreachable and denies execution, attaching an obligation to escalate to a human in the loop. The human supervisor interface presents the evidence: the ambiguous identifier, the conflicting state sources, and the contract requirement that cannot be satisfied. The supervisor chooses a safer alternative action class defined in the domain pack that can be executed under degraded mode, such as collecting additional observations or placing the asset into a safe state. Provenance records capture the denial decision, the evidence, and the human intervention.

#### Containment analysis tied to primitives:

The job contract prevented unsafe execution under degraded conditions. The policy mesh enforced contract requirements despite distributed failure. The domain pack provided an alternative constrained action path suitable for degraded modes. Provenance captured the decision trail. Monitoring later confirmed that the attempted unsafe action was blocked, and the event became an input to updating disambiguation rules in the domain pack.

This scenario demonstrates that **containment is achievable without requiring continuous human review of all actions.** It requires explicit contracts, distributed policy enforcement, and evidence capture.

## Skill misuse and capability escalation

Beyond world model errors and workflow drift, agentic enterprises face risks from skill misuse and capability escalation. These failures occur when capabilities intended for constrained tasks are recombined, chained, or repurposed into higher impact action sequences that were never explicitly authorized.

The most common driver is **overbroad capability packaging**. If domain packs bundle tools and permissions for convenience rather than explicit bounded purpose, agents gain the ability to traverse from low impact tasks into sensitive operations through intermediate steps that individually look reasonable.

Coordination further amplifies this. A lower privilege agent that cannot directly invoke a sensitive tool may still influence a higher privilege agent through requests, recommendations, or shared artifacts if the coordination fabric does not constrain influence paths by contract.

The baseline controls are **architectural**. Domain packs must be strictly scoped to purpose, with explicit separation between read access, write access, and privileged state changing actions. Job contracts must specify permitted sequences and preconditions, not only permitted tools. The policy mesh needs to enforce these constraints continuously at runtime, including enforcement on cross agent messages that attempt to trigger actions outside the sender's scope. Provenance needs to capture the full chain of capability use so that escalation paths can be detected, investigated, and blocked through targeted containment levers rather than broad shutdowns.

## Lessons from scenarios

**The key architectural lesson is that governance must be executable.** Policies and contracts must act as control plane mechanisms that bound autonomy and support safe degradation. Provenance and monitoring must be resilient so that incident response is evidence based rather than narrative based.

**The organizational lesson is that human roles must shift** toward designing, validating, and supervising the control plane and the domain packs, rather than manually checking every action.

Two practical developments are useful anchors for making this section concrete without narrowing it to a single vendor stack.

First, the **MAESTRO** framework provides an *agent-specific threat modeling lens* organized around a layered view of agentic architectures, explicitly including cross-layer interactions



and multi-agent dynamics (Huang, 2025). Even if MAESTRO is not adopted as the sole methodology, it is a clear example of how traditional threat modeling needs to be adapted when the system includes planning, memory, tool invocation, and agent-to-agent coordination.

Second, the **OWASP GenAI Security Project’s Multi-Agentic System Threat Modelling Guide** demonstrates how an *agentic threat taxonomy* can be applied to real multi-agent deployments, including explicit attention to tool misuse, intent manipulation, privilege compromise, and cascading effects across coordinated agents (OWASP GenAI Security Project, 2025). It is also notable because it treats threat modeling as an applied, architecture-aligned practice rather than a purely conceptual classification exercise.

Finally, emerging *governance control plane* proposals such as **AAGATE** show what it looks like to operationalize an AI governance framework in production by treating policy enforcement, telemetry, and accountability hooks as first-class platform capabilities rather than after-the-fact compliance artifacts (Huang et al., 2025).

The details of those proposals are less important here than the directional signal: **operational governance for agentic systems increasingly resembles reliability engineering and platform security, with continuous enforcement and measurement embedded into the runtime environment.**

### Coordination as an attack and error surface

**Coordination is both a value mechanism and a risk mechanism.** When agents exchange messages, delegate tasks, or publish intermediate findings, they create a *pathway for errors and adversarial influence to propagate*. In multi-agent systems, the most damaging failures often come from plausible intermediate artifacts that are accepted as context by downstream agents, rather than from a single obviously malicious instruction.

**This matters because coordination often crosses privilege boundaries.** A low privilege agent that cannot directly invoke a sensitive tool may still be able to send a message that causes a higher privilege agent to do so. This is why the *coordination fabric* and *job contracts* must be treated as enforceable interfaces. Messages should be typed, attributable, and policy checked. *Contracts* should constrain what kinds of requests an agent can make, what kinds of recommendations it can issue, and *which world model* partitions it can reference when coordinating with other agents.

The implications for governability are straightforward:

- **If coordination is not governed**, then even strong tool access controls can be bypassed indirectly through cross agent influence, and

- **If coordination is governed**, then investigators can trace how a downstream action was influenced and whether that influence path was permitted by contract and policy.

## Layered threat modeling, residual risks, and tradeoffs

Threat modeling for agentic enterprises **should be anchored to the substrate primitives**, not to isolated prompts or single agent flows.

The practical question is *where failures originate*, how they *propagate across* world model access, domain pack boundaries, coordination flows, and tool invocation, and *which control plane checks* would have broken the chain. This framing is compatible with the threat modeling anchors we already discussed, but the key point is that **the method can vary while the control objects remain stable**.

Even with a well-designed substrate, residual risk remains. *Some residual risk is representational*, meaning world model partitions can still be incomplete or wrong and agents can still act on stale context.

*Some residual risk is organizational*, meaning humans can misinterpret governance signals, approve actions under time pressure, or allow policy exceptions to become permanent.

*Some residual risk is systemic*, meaning coupling across agents and workflows can still produce emergent behavior that is hard to predict in advance.

The baseline posture this paper supports is not **risk eliminated**. It is **risk localized and made governable**.

The tradeoff is explicit: *higher autonomy* can increase speed and coverage while increasing the burden on the control plane to provide enforceable constraints, provenance, and fast containment levers. *Higher assurance* can reduce risk while increasing friction and requiring more explicit contract design and operational integration.

## Organizational Transformation and Workforce Design

**Architecture is necessary but insufficient.** Agentic adoption changes how work is done, how roles evolve, and how governance operates. This section provides *realistic organizational implications* while remaining *domain independent*.

### The maturity gap and pilot sprawl

Many enterprises begin with pilots. Pilots often focus on narrow productivity gains without redesigning governance. This creates a maturity gap: **agentic capability grows faster than the organization's ability to supervise it.**

Industry analysts describe organizations purchasing capability quickly while lacking measurement and governance mechanisms that demonstrate value and control (Mayer et al., 2025; Sukharevsky et al., 2025). The result is *pilot sprawl*: multiple teams deploy agents with inconsistent tool access, inconsistent logging, inconsistent policy enforcement, and unclear responsibility.

**Pilot sprawl creates structural risk.** It also creates organizational conflict because responsibility for failures becomes ambiguous. A framework backbone should therefore support portfolio level governance: domain packs, contracts, and policy mesh rules that are reusable and consistent across teams, while allowing controlled variation.

### Role evolution in hybrid human and agent organizations

In hybrid systems, **human roles shift from direct execution toward supervision, exception handling, and control plane design.** This shift is not limited to security. **It applies to any domain where agents generate actions.**

Several role patterns may emerge:

- A **domain pack engineer** designs and maintains domain packs, including tool allow lists, constraints, evaluation suites, and supervision rules. This role combines domain expertise with system design.
- A **contract and policy engineer** designs job contract templates and policy mesh logic. This role ensures that governance is executable and consistent.
- A **provenance and audit engineer** ensures evidence capture, attribution, and audit readiness. This role becomes central as agents produce artifacts that must be defensible.
- A **human supervision operator** becomes more like an air traffic controller than a traditional operator. The operator supervises multiple agent activities, intervenes at risk points, and handles exceptions. Research on remote operations and distributed

cognition highlights that AI mediated work can degrade situational awareness if interfaces and evidence are not designed to support it (Jacobsen et al., 2025). Therefore, supervision roles require tooling and training aligned with cognitive realities, not only policy statements.

- A **monitoring and evaluation lead** owns drift detection, regression testing, and scenario evaluation. This role ties runtime monitoring to governance outcomes.

These roles may be distributed across existing functions in an enterprise, but the responsibilities must exist. Without them, the organization defaults to ad hoc ownership and manual governance that will not scale.

### Responsibility assignment and accountability

**Accountability requires clarity about who owns the domain packs, the policy mesh, the agent runtime, and the outcomes of agent actions.** Accountability cannot be assigned only to *the AI team* or to *the business*. **Agentic systems are socio-technical systems.** Responsibility needs to *map to the control plane and to the delegation chain*.

*NIST AI RMF* emphasizes governance structures and accountability mechanisms (NIST, 2023). This framework adds architectural specificity. Accountability is enforced through job contracts and policy mesh decisions, and it is evidenced through provenance and attribution.

**Organizational accountability processes should therefore be aligned with these primitives**, because otherwise governance becomes disconnected from how the system actually operates.

### Workforce strategy and training implications

Workforce design needs to recognize that agents change the nature of work. Operators must learn to supervise and to interpret evidence. Engineers must learn to design domain packs and policies. Leaders must understand that “*deploying agents*” without investing in the control plane creates unmanaged risk.

External analyses argue **that workforce strategy often lags capability adoption** (Bedard, 2025). **The more general claim here is that governance capacity needs to scale with agentic capacity.** The architecture supports this by making governance work tangible: domain packs, contracts, policy rules, evaluation suites, and monitoring dashboards that reflect explicit requirements.

## Implications, Evaluation, and Future Work

This section synthesizes implications for architecture and governance, then describes evaluation priorities and future work directions that can be pursued in domain instantiation papers.

### Architectural implications

**If agentic systems are treated as minor extensions of existing automation, organizations will inherit unmanaged autonomy drift, inconsistent governance, and attribution failure.**

The proposed framework implies that governable agentic deployment requires a control plane substrate with explicit primitives. World model handling, domain packs, job contracts, policy mesh enforcement, provenance, attribution, and monitoring form a coherent set. *Removing one weakens the rest.*

For example, policies without provenance create opaque enforcement. Provenance without contracts creates logs without bounding. Monitoring without explicit requirements becomes mere telemetry.

The framework also implies that distributed systems constraints needs to be treated as first class. Degraded mode policies and distributed enforcement are required for continuity. Centralized manual governance *cannot survive* partitions without either halting operations or being bypassed.

### Ethical accountability and responsibility

**A governable agentic architecture does not remove accountability; it makes accountability explicit.**

The core principle is that responsibility for outcomes cannot be delegated to an agent, even when execution is delegated. Enterprises remain accountable for the decisions that agents make under enterprise authorization, and that accountability needs to be traceable through the same substrate primitives that bound autonomy.

In this framework, ethical accountability is enforced through explicit ownership and explicit evidence. Every domain pack, job contract, and world model partition needs to have a named human owner, a governance authority, and a change control path. When an agent acts, the system must be able to reconstruct not only what happened, but also who authorized the capability, who approved the constraints, what policy checks were in effect, and what evidence was produced at the time of decision. Without those linkages,

“accountability” collapses into post hoc narrative, because responsibility becomes socially asserted rather than technically reconstructable.

This is also where supervision is defined precisely. Human oversight in agentic systems cannot be reduced to “*a human was in the loop somewhere*.” Oversight has to be described as a set of supervision modes tied to risk. Some actions can be allowed under continuous policy enforcement and monitoring, while others require explicit human approval or an enforced safe mode when required checks cannot be performed. Ethical accountability therefore becomes a control plane property: the system needs to prevent high impact actions from being executed under degraded evidence conditions, and it needs to expose the reasons when it denies, escalates, or constrains autonomy.

Finally, ethical accountability requires clear constraints on cross boundary influence. Agents that can coordinate across teams, systems, and privileges can unintentionally create pressure pathways where a low privilege agent influences a high privilege agent to take an action that would not be permitted directly. The coordination fabric and policy mesh should therefore encode *who can ask what of whom* as an auditable rule set, not a social convention. The goal is not to eliminate error or harm, but to ensure that when errors occur, the enterprise can identify the authorized delegation chain, the enforced constraints, and the exact points where governance failed or was bypassed.

## Limitations

This paper proposes a reference architecture and a set of primitives intended to make agentic behavior governable at enterprise scale. It does not claim that these primitives are the only possible representation, nor does it claim that architecture alone guarantees trustworthy outcomes. The claim is narrower: without something functionally equivalent to governed world model handling, bounded capability packaging, enforceable job contracts, and continuous policy enforcement, enterprise governance goals such as auditability, accountability, and controllable behavior become structurally infeasible as autonomy scales.

The framework is also intentionally domain independent. That is a strength for portability, but it creates limitations in prescriptiveness. The paper does not provide sector specific control mappings, regulatory interpretations, or detailed implementation patterns for any single domain such as healthcare, finance, safety critical robotics, or national security environments. Those require follow on work that instantiates the primitives into concrete control objectives, operational processes, and assurance evidence suitable for the domain.

In addition, the scenarios and requirements here focus on runtime governability rather than the full end to end lifecycle of model development. Data curation, training security, model supply chain integrity, and the empirical measurement of model capabilities are important topics, but they are not the primary scope of this architecture. This paper assumes that

organizations will continue to manage those lifecycle concerns, while arguing that runtime governability and attribution require their own substrate.

Finally, evaluation in agentic contexts remains an open engineering and research problem. The paper proposes evaluation as an architectural requirement, but the field lacks standardized benchmarks that map cleanly to enterprise governance outcomes. A central limitation is therefore that many claims about effectiveness depend on future empirical validation and on the emergence of shared evaluation practices across organizations.

### Evaluation priorities

Evaluation must focus on governance properties, not only task success.

Several evaluation axes are implied by the framework:

- contract compliance rates,
- policy violation attempts and outcomes,
- provenance completeness under failure,
- drift detection sensitivity and false positive rates,
- human supervision workload and situational awareness outcomes, and
- containment effectiveness in scenario tests.

Human factors research shows that increasing automation changes human workload and situational awareness in complex ways (Parasuraman, Sheridan, and Wickens, 2000; Endsley, 2017). Operational research in security shows that explanation mechanisms can increase cognitive burden if not aligned with decision needs (Rastogi et al., 2025; Tariq et al., 2025). Therefore, evaluation needs to include human supervision performance metrics, not only system performance metrics.

### Proof of concept validation approach

A proof of concept for this architecture should not attempt to “*prove the whole framework*” at once. It should validate the governability claims using a constrained workflow where governance failures are easy to observe and where containment and reconstruction have clear operational meaning. The objective is to demonstrate that the primitives create measurable improvements in attribution, containment, and policy enforceability under realistic partial failure conditions.

At minimum, a proof of concept should implement one bounded workflow with multiple agents, tools, and intermediate artifacts, and it should instantiate the substrate explicitly: a world model partition for the workflow, a domain pack that defines the allowed capabilities, a job contract that encodes explicit behavioral constraints, and a policy mesh that mediates access to tools and data. Provenance and attribution should be captured by default for each

action and artifact, and monitoring should evaluate observed behavior against the job contract and policy expectations rather than only logging tool calls.

Validation should then focus on measurable questions aligned to enterprise governance, not model performance.

The proof point is whether the system can

- a) prevent or constrain unsafe actions under defined policy,
- b) degrade safely when required checks cannot be performed,
- c) reconstruct the delegation chain and decision context during incident response, and
- d) localize which contract, pack, policy, or world model assumption needs to change to remediate a class of failures.

A meaningful proof of concept should include at least one injected partial failure case, such as an unavailable validation dependency or a logging degradation event and show that the architecture either blocks high impact actions or forces an explicit degraded mode that remains observable and attributable.

### Future work directions

Future work can refine each primitive and its mapping into domain papers.

**World model governance** needs stronger methods for detecting stale or contradictory context across distributed systems and for representing uncertainty in ways that support safe planning.

**Domain pack engineering** needs methods for building reusable, testable constraint bundles across domains, including embodied domains where safety constraints are physical.

**Policy mesh design** needs patterns for distributed enforcement under partition, including proofs or empirical evidence for safe degraded modes.

**Provenance and attribution** need standardized schemas that can capture agent delegation chains and tool calls across heterogeneous systems, aligned with provenance models and content authenticity standards where applicable (W3C, 2013; C2PA, 2024).

**Monitoring** needs robust drift detection and anomaly detection tied to explicit contracts and policies, and evaluation suites that reflect real operational risks.

**Organizational research** is needed on how supervision roles evolve and how enterprises can sustain governance capacity as agent fleets scale. Research on distributed cognition in AI mediated operations provides a starting point (Hutchins, 1995; Jacobsen et al., 2025).



## Research agenda and standardization directions

**The architecture identifies a functional substrate for governable agency, but it also exposes where the ecosystem is immature.** Several research and standardization directions follow naturally. Formal languages for job contracts are needed so contracts can be expressed, validated, and enforced consistently. Representation standards are needed for domain packs, including capability schemas, policy hooks, and world model bindings, so that packs can be shared, audited, and versioned. Methods for quantifying and localizing world model drift are needed so representational changes can be tied to changes in agent behavior. Identity and policy models that treat agents and humans consistently across policy meshes require refinement so authorization, delegation, and accountability are coherent across mixed human and agent organizations.

In parallel, evaluation needs a shared empirical footing. The field needs benchmarks and measurement methods that connect agent behavior to enterprise outcomes such as containment time, policy violation rates, provenance completeness, and reconstruction accuracy under partial failure. Without that, *governability* risks becoming a purely qualitative claim.

Standardization efforts in AI governance and security are beginning to incorporate agentic concerns, but most are still organized around *lifecycle process guidance* rather than *enforceable runtime primitives*.

**This architecture can help structure that work by giving standards bodies and enterprises a concrete set of objects, events, and interfaces to standardize around.**

## The Need for New Governance Mechanisms

**Agentic enterprises require governance mechanisms that match the tempo and scale of agent action generation.** Manual governance does not scale because of cognitive limits, action volume, and distributed systems constraints. A domain independent framework can remain high level while still being operationally sharp by defining stable primitives and design requirements: world model, domain pack, job contract, policy mesh, provenance, attribution, and monitoring. These primitives enable follow on domain instantiation papers to specify concrete controls, artifacts, and evaluation methods without inventing new governance concepts.

**The goal is not to slow adoption. The goal is to make adoption governable.**

A final implication is that audit and investigation in agentic systems must shift from *what did the code do* to *what decision lineage occurred*.

This requires *capturing not only outputs but the chain of tool calls, retrieved context, memory writes and reads, policy checks, and the boundaries of autonomy that were in effect at the*

*time*. Without that lineage, incident response becomes *narrative reconstruction* rather than *evidence-based analysis*.

### From Hype to Measured Autonomy

This paper started from a simple premise. Agentic systems are arriving as modular fleets, not as single monolithic models. That shift changes the problem space. The question is no longer whether a model can answer a prompt. The question is whether a distributed system of agents can sustain intent, execute reliably across tools and time, and remain legible enough that an enterprise can operate it without drowning in supervision, rework, and exception handling.

The empirical record now available forces a disciplined position. Autonomous enterprises are not imminent as a general claim. Across enterprise-analog benchmarks, web and OS interaction suites, tool-use evaluations, and real repository software engineering benchmarks, agents repeatedly demonstrate competence in localized execution while failing to reliably carry multi-step tasks through to completion in stateful environments. The failure modes are not exotic. They are operational. Agents lose task intent, mis-handle interfaces, struggle to recover from minor deviations, and sometimes produce outputs that look complete but are not. This is the core boundary condition. Execution reliability, not fluency, is the limiting factor for near-term autonomy.

That boundary condition does not weaken the case for agents. It sharpens it. It moves the conversation away from replacement fantasies and toward architecture. If agents are unevenly reliable, then the enterprise value proposition is not autonomous substitution. It is throughput amplification inside bounded work envelopes. The near-term winning pattern is orchestration: agents embedded inside human-led workflows where the system constrains scope, checkpoints progress, preserves state, and routes exceptions cleanly. Fleets can be used to parallelize well-defined work, absorb spikes, and accelerate cycles. They should not be assumed capable of replacing the organizational core functions that depend on ambiguity resolution, social coordination, and exception-driven judgment.

The deeper implication is that autonomy is as much an environmental property as it is a model property. Many enterprise workflows are difficult for agents not because they are cognitively complex, but because they are underspecified, UI-bound, and full of hidden dependencies. If organizations want higher autonomy, they will need to redesign work so that it becomes machine-legible. That means explicit success criteria, fewer opaque UI-only states, more structured interfaces, observable checkpoints, and workflows that can be resumed without semantic loss. Without that shift, gains in model capability will increase

local performance while the enterprise-level autonomy ceiling remains pinned by the messiest surfaces.

This paper's core contribution is therefore a governability framing for agentic enterprises. The path forward is not to wait for a single breakthrough model that makes the problem disappear. The path forward is to build systems that can safely and predictably operate within known capability limits, then progressively raise autonomy by improving both the agent and the environment in which it works. Autonomy will rise, but it will rise along the constraints empirical benchmarks are already exposing, not along marketing narratives.

What follows this conclusion is a clear standard for enterprise deployment maturity. A system is not *enterprise ready* because it can demonstrate isolated successes.

**It is *enterprise ready* when it can sustain *execution over time*, *surface truthful state*, *recover from failure without losing meaning*, and *keep the human burden of verification proportional to the value gained*.**

**That is the practical definition of governability.**

If enterprises adopt that definition, they can capture the benefits of agent fleets now while building a credible pathway toward higher autonomy over time, without betting organizational stability on wishful thinking.

## References

- Anne, M., et al. (2024). *Harnessing language for coordination: A framework and benchmark for LLM-driven multi-agent control (HIVE)*. arXiv. <https://arxiv.org/abs/2412.11761>
- Bedard, J., Lucero, E., Ebeling, R., Breitling, F., Garcia-Garcia, C., & Sakhuja, A. (2025). *AI is moving faster than your workforce strategy*. Boston Consulting Group. <https://www.bcg.com/publications/2025/ai-is-moving-faster-than-your-workforce-strategy>
- Coalition for Content Provenance and Authenticity. (2024). *C2PA technical specification* (Version 2.1). [https://spec.c2pa.org/specifications/specifications/2.1/specs/\\_attachments/C2PA\\_Specification.pdf](https://spec.c2pa.org/specifications/specifications/2.1/specs/_attachments/C2PA_Specification.pdf)
- Endsley, M. R. (2017). From here to autonomy: Lessons learned from human-automation research. *Human Factors*, 59(1), 5–27. <https://doi.org/10.1177/0018720816681350>
- Fung, P., Bachrach, Y., Celikyilmaz, A., Chaudhuri, K., Chen, D., et al. (2025). *Embodied AI agents: Modeling the world*. arXiv. <https://doi.org/10.48550/arXiv.2506.22355>
- Gilbert, S., & Lynch, N. (2002). Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2), 51–59. <https://doi.org/10.1145/564585.564601>
- Ha, D., & Schmidhuber, J. (2018). *World models*. arXiv. <https://doi.org/10.48550/arXiv.1803.10122>
- Hu, Y., Gao, Y., & An, B. (2014). Multiagent reinforcement learning with unshared value functions. *IEEE Transactions on Cybernetics*, 45(4), 647–662. <https://doi.org/10.1109/TCYB.2014.2332042>
- Huang, K. (2025, February 6). *Agentic AI threat modeling framework: MAESTRO*. Cloud Security Alliance. <https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro>
- Huang, K., Lambros, K. R., Huang, J., Mehmood, Y., Atta, H., Beck, J., Narajala, V. S., Baig, M. Z., Ul Haq, M. A., Shahzad, N., & Gupta, B. (2025). *AAGATE: A NIST AI RMF-aligned governance platform for agentic AI*. arXiv. <https://doi.org/10.48550/arXiv.2510.25863>
- Hutchins, E. (1995). *Cognition in the wild*. MIT Press. <https://mitpress.mit.edu/9780262082310/cognition-in-the-wild/>
- ISO/IEC. (2023a). *ISO/IEC 23894:2023: Information technology—Artificial intelligence—Guidance on risk management*. International Organization for Standardization. <https://www.iso.org/standard/77304.html>
- ISO/IEC. (2023b). *ISO/IEC 42001:2023: Information technology—Artificial intelligence—Management system*. International Organization for Standardization. <https://www.iso.org/standard/81230.html>

- Jacobsen, R. M., Wester, J., Djernæs, H. B., & van Berkel, N. (2025). *Distributed cognition for AI-supported remote operations: Challenges and research directions*. arXiv. <https://doi.org/10.48550/arXiv.2504.14996>
- Jimenez, C., et al. (2024). *SWE-bench: Can language models resolve real-world GitHub issues?* In *Proceedings of the International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=VTF8yNQM66>
- Jin, W., Du, H., Zhao, B., Tian, X., Shi, B., & Yang, G. (2025). *A comprehensive survey on multi-agent cooperative decision-making: Scenarios, approaches, challenges and perspectives*. arXiv. <https://doi.org/10.48550/arXiv.2503.13415>
- Li, Z., et al. (2024). *DevBench: A comprehensive benchmark for software development*. arXiv. <https://arxiv.org/abs/2403.08604>
- Lu, Y., et al. (2025). *Exploring autonomous agents: A closer look at why they fail when completing tasks*. arXiv. <https://arxiv.org/abs/2508.13143>
- Ma, Y., et al. (2024). *M&M's: A benchmark to evaluate tool-use for multi-step multi-modal tasks*. arXiv. <https://arxiv.org/abs/2403.11085>
- Mayer, H., Yee, L., Chui, M., & Roberts, R. (2025, January). *Superagency in the workplace: Empowering people to unlock AI's full potential at work*. McKinsey & Company. [https://www.mckinsey.com/~/\\_media/mckinsey/business%20functions/quantumblack/our%20insights/superagency%20in%20the%20workplace%20empowering%20people%20to%20unlock%20ais%20full%20potential%20at%20work/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-v4.pdf](https://www.mckinsey.com/~/_media/mckinsey/business%20functions/quantumblack/our%20insights/superagency%20in%20the%20workplace%20empowering%20people%20to%20unlock%20ais%20full%20potential%20at%20work/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-v4.pdf)
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2), 81–97. <https://doi.org/10.1037/h0043158>
- Moreau, L., & Missier, P. (Eds.). (2013, April 30). *PROV-DM: The PROV data model* (W3C Recommendation). World Wide Web Consortium. <https://www.w3.org/TR/prov-dm/>
- National Institute of Standards and Technology. (2020). *Zero trust architecture* (NIST SP 800-207). <https://doi.org/10.6028/NIST.SP.800-207>
- National Institute of Standards and Technology. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* (NIST AI 100-1). <https://doi.org/10.6028/NIST.AI.100-1>
- National Institute of Standards and Technology. (2024). *Artificial intelligence risk management framework: Profile for generative AI* (NIST AI 600-1). <https://doi.org/10.6028/NIST.AI.600-1>
- OWASP GenAI Security Project. (2025). *Multi-agentic system threat modeling guide* (Version 1.0). <https://genai.owasp.org/resource/multi-agentic-system-threat-modeling-guide-v1-0/>

- Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 30(3), 286–297. <https://doi.org/10.1109/3468.844354>
- Qian, C., et al. (2025). *Scaling large language model-based multi-agent collaboration*. In *Proceedings of the International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=K3n5jPkrU6>
- Qiu, R.-Z., Song, Y., Peng, X., Suryadevara, S. A., Yang, G., Liu, M., Ji, M., Jia, C., Yang, R., Zou, X., & Wang, X. (2024). *WildLMA: Long-horizon loco-manipulation in the wild*. arXiv. <https://doi.org/10.48550/arXiv.2411.15131>
- Rastogi, N., et al. (2025). Too much to trust? Measuring the security and cognitive impacts of explainability in AI-driven SOC. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. <https://arxiv.org/abs/2503.02065>
- Raza, S., Sapkota, A., Karkee, P., & Emmanouilidis, C. (2025). *Toward security risk management for agentic AI systems*. TechRxiv. <https://doi.org/10.36227/techrxiv.175735299.97215847>
- Su, H., Luo, J., Liu, C., Yang, X., Zhang, Y., Dong, Y., & Zhu, J. (2025). *A survey on autonomy-induced security risks in large model-based agents*. arXiv. <https://doi.org/10.48550/arXiv.2506.23844>
- Sukharevsky, A., Kerr, D., Hjartar, K., Hämäläinen, L., Bout, S., Di Leo, V., & Dagorret, G. (2025, June). *Seizing the agentic AI advantage: A CEO playbook to solve the gen AI paradox and unlock scalable impact with AI agents*. McKinsey & Company. <https://www.mckinsey.com/~media/mckinsey/business%20functions/quantumblack/our%20insights/seizing%20the%20agentic%20ai%20advantage/seizing-the-agentic-ai-advantage.pdf>
- Tariq, S., Chhetri, M. B., Nepal, S., & Paris, C. (2025). Alert fatigue in security operations centres: Research challenges and opportunities. *ACM Computing Surveys*, 58(8), Article 176. <https://doi.org/10.1145/3723158>
- Xu, Y., et al. (2024). *TheAgentCompany: Benchmarking LLM agents on consequential real-world tasks*. arXiv. <https://arxiv.org/abs/2412.14161>
- Xie, T., et al. (2024). *OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments*. arXiv. <https://arxiv.org/abs/2404.07972>
- Yao, S., et al. (2024).  *$\tau$ -bench: A benchmark for tool-agent-user interaction in real-world domains*. arXiv. <https://arxiv.org/abs/2406.12045>
- Yoran, O., et al. (2024). AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. <https://aclanthology.org/2024.emnlp-main.505/>