DOME SYSTEMS

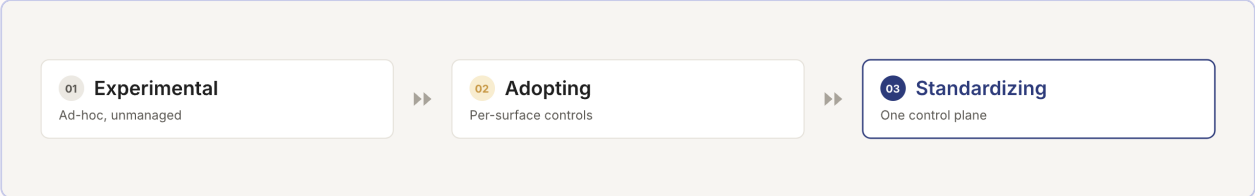
Toward Enterprise Agentic Operations

A maturity model for agentic adoption

Introduction

Enterprise AI adoption looks chaotic from the outside, and it often is on the inside too. A first team builds something with Claude Code. Another stands up an agent on LangGraph. Finance signs an enterprise contract with a writing assistant. Sales rolls out a meeting-summarization tool. Individual employees are running things on the side that nobody has been told about. None of this is wrong. All of it is producing value. And none of it is being governed in any coherent way.

The temptation in this moment is to manage by reflex. Block what looks risky. Sponsor what looks promising. Add a tool when a new problem appears. The instinct is right — something needs to be done — but the response is unstructured, and unstructured responses to a structural shift compound in cost over time.



There is a structure underneath what looks like chaos. The state of an enterprise's agent operations falls into one of three stages — Experimental, Adopting, Standardizing — and each stage has a recognizable pattern, predictable costs, and a recognizable trigger to the next. Knowing where you are makes the next move obvious. Knowing where you are going makes the current investment legible.

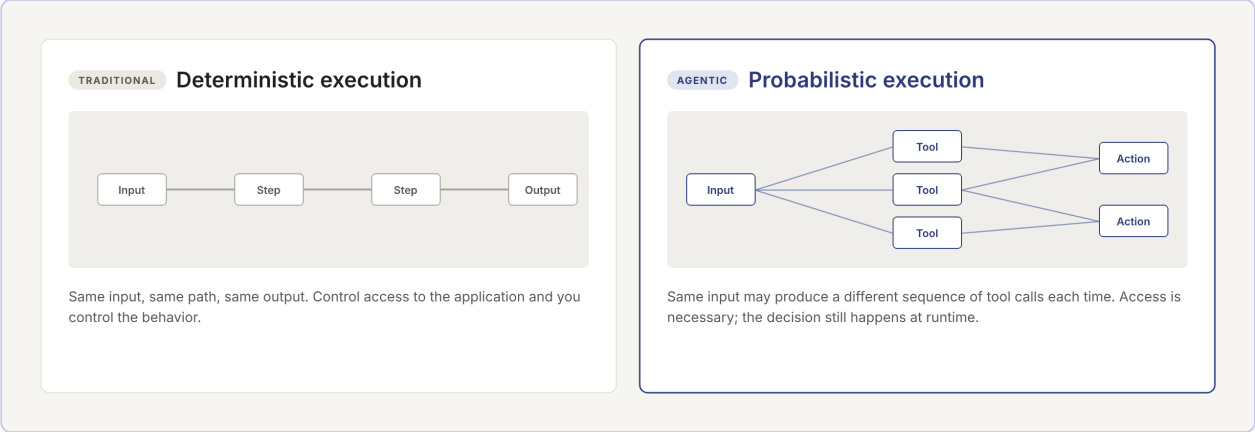
This paper is for the people deciding to invest in enterprise agentic operations — executive sponsors, platform leaders, and security leaders setting direction for the next several years. The choices made now shape how the enterprise governs (and benefits from) agents at scale.

What we mean by an agent

Regardless of how it is built, where it runs, or who supplied it, every agent is composed of three things: **Code** (the runtime that orchestrates the agent), **Models** (the reasoning engine that decides what to do next), and **Tools** (anything the agent calls to act in the world — APIs, databases, messaging, MCP servers). Governance has to attach to each of these primitives. Frameworks change every quarter; the primitives don't.

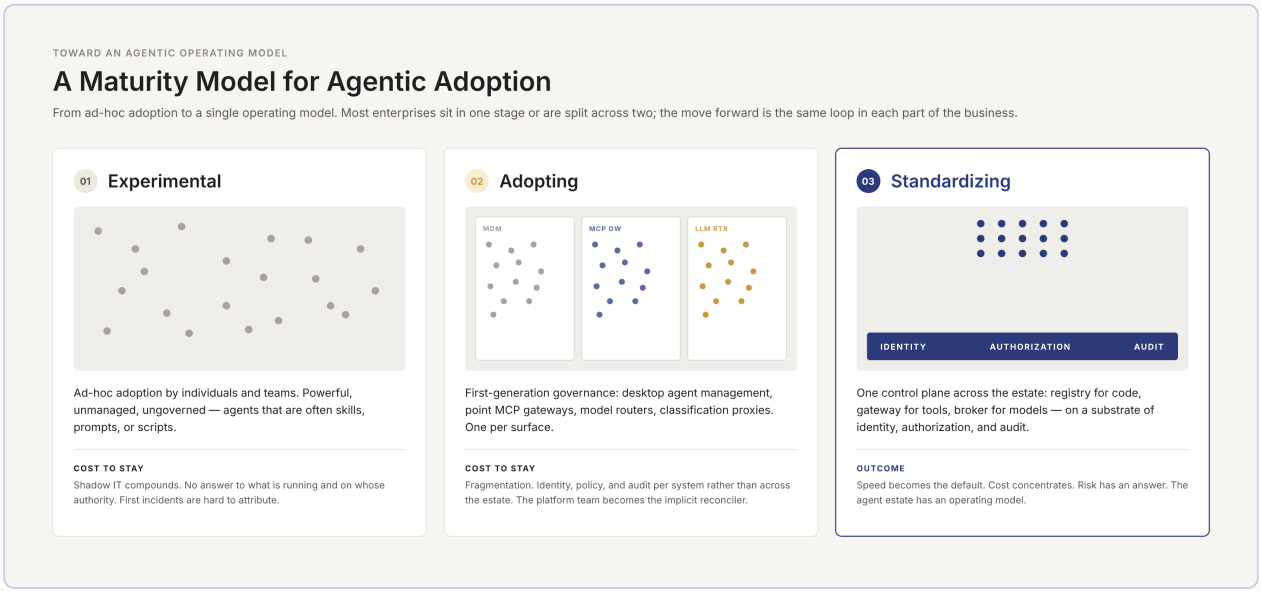
Three properties distinguish the agent estate from anything platform teams have governed before:

- **Probabilistic execution.** Agents reason at runtime. The same input may produce a different sequence of calls each time. Existing governance assumes deterministic paths.
- **Multi-surface deployment.** Agents appear inside cloud platforms, data platforms, SaaS applications, vendor desktop tools, and on individual developer machines, simultaneously. No single perimeter contains the estate.
- **Faster than the response.** A working agent prototype is one weekend's work. The pace of agent appearance routinely outruns the pace at which any one team can wrap policy around them.



The three stages of agentic maturity

As an approximation, an enterprise's agent operations can be grouped along a path from ad-hoc to standardized. A team and organization is usually somewhere on the path in one part of the business and somewhere different in another. We've classified the stages below — Experimental, Adopting, Standardizing — as reference points along that path, not boxes the organization fits into.



Stage 1 — Experimental

Agents are being built ad hoc by individuals and teams.

The defining feature of the Experimental stage is the absence of a posture. Some agents are explicitly sanctioned (a pilot in the data team, a sponsored coding tool); most appear because somebody on a Tuesday afternoon found a problem and a model that could solve it before Wednesday morning.

What is running, in practice:

- Developers using AI coding assistants with personal API keys, against repositories that contain production credentials
- Business teams running agents inside SaaS applications they bought without IT awareness
- Ad-hoc scripts and workflows that connect a model to a corporate API through a single shared key

- Vendor desktop tools (Claude Desktop, Cursor, Copilot, MCP-driven assistants) configured by the user, governed by nothing on the enterprise side

The "agents" are often something simpler: a prompt, a script, a skill. They touch real data, hold real privileges, and generate real exposure. The Experimental stage is what the new shadow IT looks like in 2026.

What the stage produces. Learning, fast. This is where the organization discovers what agents are good for, where workflows are ripe for automation, and which teams have the appetite to invest. It is not a stage to skip or suppress; it is the source of the demand that justifies the investment in everything that follows.

What it costs to stay. Discoverability collapses — nobody can produce a current answer to which agents are running, what they can reach, and who is accountable. Incidents are unattributable — when something goes wrong, the forensic trail is whatever the underlying systems happened to log, and it is rarely enough. And eventually the deal that needs an answer arrives: a customer security questionnaire asks about agent governance, and there is no consolidated answer the enterprise can give.

The trigger to move. Three triggers, in roughly increasing order of severity: a CISO with budget, a near-miss incident, or a regulator asking. In every case the move is the same — bring at least one surface under management, based on the most acute risk.

Stage 2 — Adopting

Governance arrives, one surface at a time.

In the Adopting stage, the organization has accepted that agents are real and that governance is a first-class concern. The response is to acquire or build governance for specific surfaces. This is real progress, and it is where most of the market sits today.

What this typically looks like:

- A **desktop agent management** solution for vendor tools, often through MDM or an endpoint policy product
- An **MCP gateway** to put policy and audit in front of one or two MCP server fleets
- An **LLM router or proxy** to consolidate model spend and apply per-team quotas
- A **data classification or DLP proxy** to redact sensitive content moving toward AI tools

Each of these is a real control. Each solves a real problem. The platform and security teams are doing exactly what they should: addressing the most acute risks with the most appropriate tool.

What the stage produces. Localized governance. The desktop assistant fleet is auditable. Model spend has ceilings. The MCP servers serving the sales team have policy on what they

expose. The CISO can answer narrower questions with more confidence than they could six months earlier.

What it costs to stay. Fragmentation. It is rarely visible in any given period — each individual control feels like progress — but it becomes the dominant cost over the multi-year horizon. Four recurring symptoms:

1. **Identity does not propagate.** The MCP gateway knows the agent. The model router knows the team. The DLP proxy knows the user. Nothing knows all three.
2. **Audit is per system.** Each control emits its own events in its own format. Reconstructing an incident means correlating three or four feeds with different identifiers and time windows.
3. **Policy is per language and per system.** A rule written for the model router does not apply to the gateway. A rule for the gateway does not constrain what the agent does inside a SaaS product.
4. **Surfaces appear faster than tools.** A new ISV-supplied agent, a new data-platform-embedded agent, a new internal team that built its own runtime — each one requires another tool to govern, or remains ungoverned.

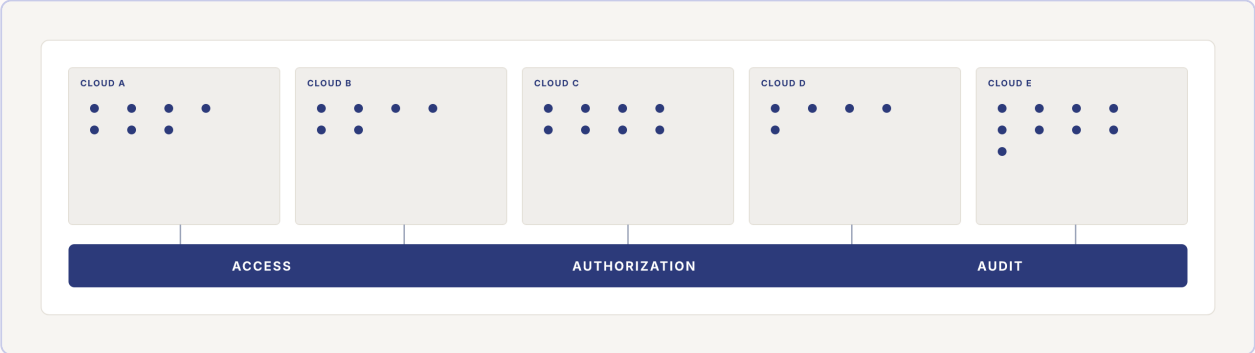
The Adopting stage is where most enterprise agent strategies look reasonable in isolation and unreasonable in aggregate. The platform team is doing the right work, with the right intent, and accumulating a substrate that is harder to operate every quarter.

The trigger to move. The Adopting-to-Standardizing trigger is usually a recognition rather than an incident. The platform team realizes the stitching is the system, and the system cannot scale at the pace agents are appearing. Often this is precipitated by a strategic question — we want to ship agents to our customers — that the current substrate cannot answer without another six months of integration.

Stage 3 — Standardizing

Every agent is governed against one substrate.

The Standardizing stage is what enterprise agentic operations look like in practice. One control plane sits underneath the agent estate. Every agent, regardless of framework, model, or runtime, is governed against the same access, policy substrate, and audit stream. The fragmentation cost of the Adopting stage falls toward zero, because there is one substrate to operate instead of five.



At this stage the operating model is shaped around the agent unit. Three control points correspond to the three primitives, and a shared substrate delivers the outcomes governance teams actually consume.

Control point	Governs	Outcome
Agent Registry	Code	Every agent is registered with a capability declaration, and a managed lifecycle
Tool Gateway	Tool calls	Every external action is mediated; policy is evaluated at the call boundary; credentials are injected at egress, never held by the agent
Model Broker	Model calls	Every model call passes through one routing point; provider selection, classification, and spend are governed centrally

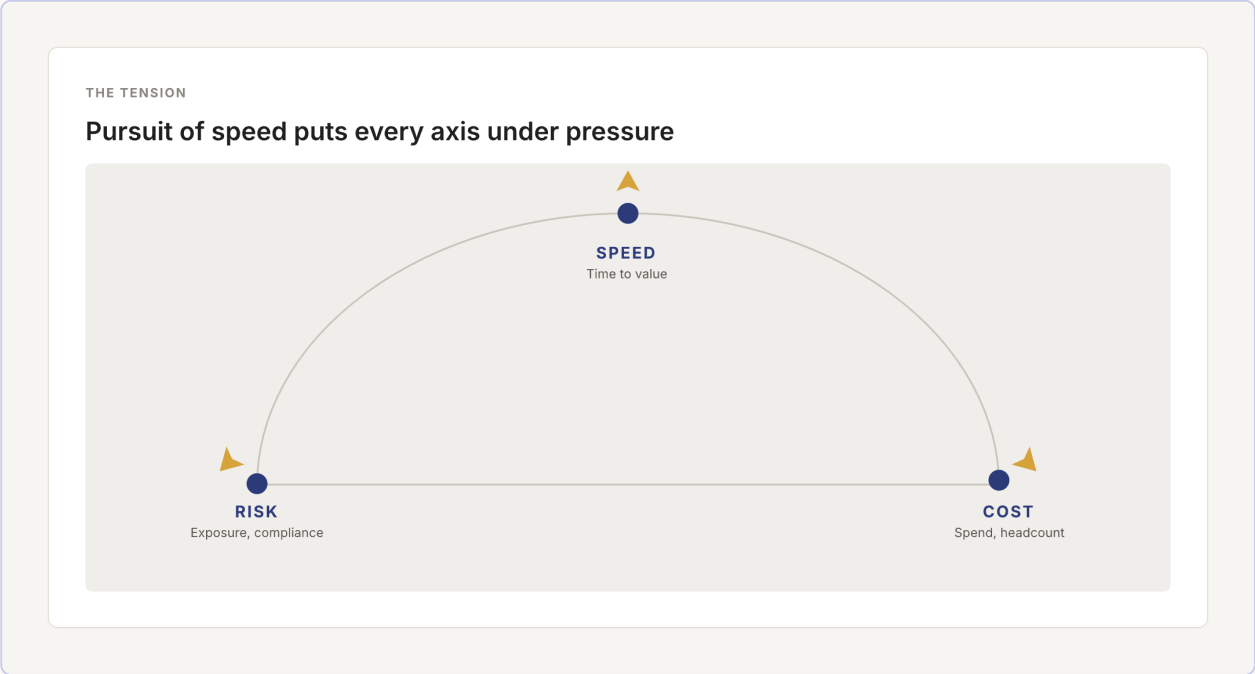
Substrate	What it provides
Authorization	One policy model evaluated at org, tenant, workspace, and agent scopes; forbid wins; deterministic decisions
Audit	One immutable stream covering registry, gateway, and broker activity, queryable by agent, user, tool, model, or policy

This is the shape Dome implements, and it is the shape any standardized agent operating model converges on. The control points compose: an agent in the Registry, calling tools through the Gateway, against models served by the Broker, is fully under management — and every decision becomes a row in a single audit stream.

Standardizing is not an endpoint; it is an operating model that admits ongoing refinement. The work after the platform is in place is coverage breadth (every surface), policy depth (org baselines, tenant refinements, agent constraints), and operational autonomy (templates, tier ladders, declarative onboarding, self-service for teams). The maturity continues; the substrate is no longer the constraint.

What standardizing actually changes

Every prior wave of enterprise digital transformation has produced the same dynamic: the new capability accelerates speed, and that pursuit of speed creates corresponding pressure on cost and risk. Cloud did it. Containers did it. Mobile did it. AI is doing it now, and at higher amplitude — because agents act on data and tools, not just render pages.



The three stages of maturity can be read as a story about controls. They are also a story about the underlying enterprise economics — the framing that matters most for the people approving the budget. Standardizing exists because it is the only stage that resolves the tension: it lets the enterprise pursue speed without absorbing the cost and risk consequences.

Speed — yes becomes the default. Every new agent inherits the substrate. The platform team becomes the source of yes, not the source of friction. Teams self-serve within constraints the platform team knows are safe.

Cost — governance scales automatically. Once the platform is in place, every new agent inherits it. The marginal cost is a policy review, not an integration project — headcount no longer grows with the estate. Model and tool spend consolidate.

> **Risk — attribution replaces reconstruction.** Every governed action is recorded with full context. The chain reconstructs itself. Compliance evidence becomes a query, not a project.

Adopting-stage governance scales with people. Each new agent surface needs someone to integrate the latest gateway, reconcile the latest audit format, manage another vendor relationship — and the cost of governance grows with the size of the estate. Standardizing inverts this: governance is built into the substrate; new agents inherit it. Over the multi-year horizon, that is the difference between a governance function that grows linearly and one that grows logarithmically.

The same inversion applies to incidents. In Adopting, an incident is reconstructed after the fact from whatever logs happened to exist — a best-effort narrative with gaps where the relevant system did not log enough. In Standardizing, the audit stream is the artifact. Every governed action is recorded with full context: which agent, on whose behalf, against which rule, with what outcome. Regulators are converging on the position that decisions made by AI systems must be explainable, auditable, and reproducible. The Adopting stage cannot reliably produce any of the three across the estate. The Standardizing stage produces all three as a property of the substrate.

Measuring the outcome

Each outcome has signals that distinguish Adopting from Standardizing — concrete enough to track in a quarterly review.

Outcome	What to measure	Adopting	Standardizing
Speed	Time to onboard a new agent	Days to weeks (per-surface integration)	Hours (template plus policy review)
Speed	Share of new agents self-served by teams	Under 20%	Over 80% within guardrails
Cost	Governance headcount as estate grows	Linear with estate size	Logarithmic; plateaus
Cost	Model and tool spend strategy	Per-team negotiations, fragmented	One vendor strategy across the estate
Risk	Mean time to attribute an incident	Hours to days (correlate across systems)	Minutes (single audit query)
Risk	Audit-evidence completeness	Best-effort reconstruction	Every governed action recorded with full context

Mapping your own stage

A short diagnostic, applied per part of the organization, places you accurately.

Question	Experimental	Adopting	Standardizing
Can someone produce a current list of agents running in production?	Not without surveying teams	Yes, for sponsored agents; not for the long tail	Yes, as a property of the registry
If an incident happened today, could you reconstruct what the agent did, on whose behalf, and under what policy?	From application logs, mostly	By correlating across two or three systems	From a single audit query
How is a new agent given access to a new tool?	The team that builds the agent figures it out	Through a per-surface workflow, usually a ticket	Self-service against policy, within guardrails
If a regulator asked you to demonstrate that agent decisions are made within policy, what would you show them?	The agent's source code	A combination of policy documents and per-system logs	The policy bundle, the audit stream, and the simulation history

Most enterprises are split across stages. The work is not to collapse all of it onto one timeline. It is to choose one substrate and let coverage expand from where it makes the most economic sense to start.

A path forward

Whatever the current stage, the move toward Standardizing is the same loop. Start with the control point that maps to the most acute risk, adopt it for one agent end-to-end, then repeat.

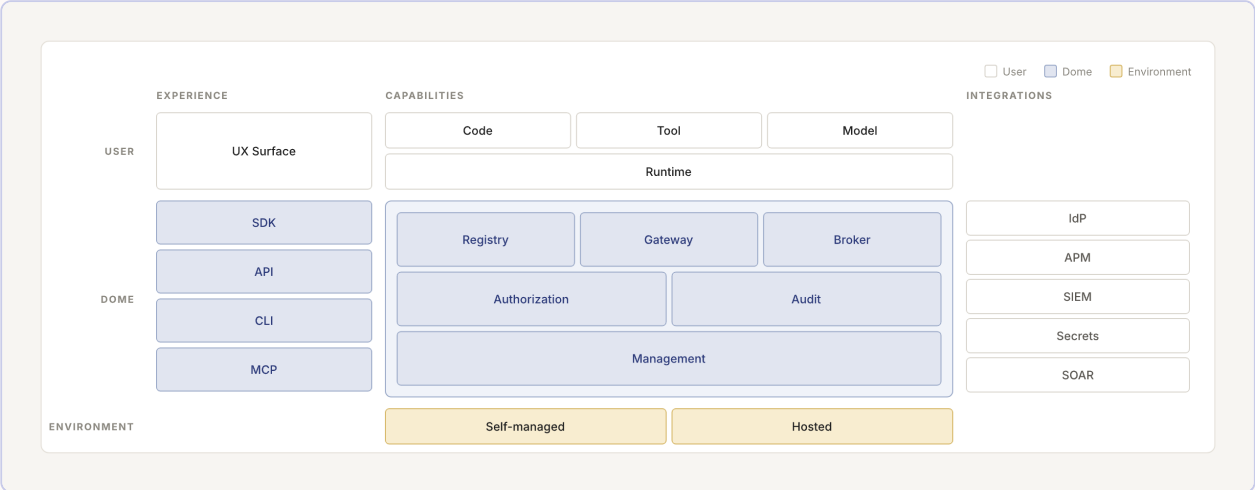
If the felt risk is...	Start with...	Because it gives you...
"We don't know what agents are running"	**Registry (Code)**	A canonical inventory and lifecycle controls
"Agents are calling tools we can't see or constrain"	**Gateway (Tools)**	Policy on every external call, with credential isolation
"Model spend and data egress are out of control"	**Broker (Models)**	One enforcement point for model traffic, with per-call audit

The loop on each control point is recognizable from any other platform rollout: register, scope, wire, author policy, simulate, activate, observe. The first agent through the loop will take longer than expected. The tenth will be a fraction of the time. The hundredth will be a template the team that owns the agent runs without the platform team in the room.

The three are stronger together, and each works alone. The full value comes from running all three control points on one substrate — identity propagates end-to-end, audit is uniform, policy composes across them. The platform is also designed so each can be adopted on its own when that matches where the enterprise is: a Registry-first deployment to build inventory, a Gateway-first deployment to bring tool calls under policy, a Broker-first deployment to govern model spend. Coverage compounds as the others come online.

The Dome Platform

Dome is an agentic operations platform — a system of control that provides consistent governance across every agent, every runtime, every cloud. The shape is the same regardless of where the enterprise sits on the maturity curve: the agents the business is already building meet a single Dome layer that the platform team operates, on top of the systems already in production.



Above Dome — the agents you already build. Code, tools, and models, running in whatever runtime the team chose. Nothing about how an agent is built changes when it comes under management.

Inside Dome — the platform itself. Dome meets developers and operators through the surfaces they already prefer (SDK, API, CLI, MCP), implements the three control points (Registry for code, Gateway for tools, Broker for models) on a substrate of Authorization and Audit, and gives platform teams a Management layer to administer the org, tenants, and workspaces underneath all of it.

Below Dome — the systems you already run. The platform integrates outward with what's already in production: identity providers, observability, SIEMs, secrets managers, automation. Deployment is self-managed or hosted, matched to the data-residency and operating posture of the enterprise.

Explore the platform, browse solutions by pattern and persona, or get started.

Conclusion

Every era of enterprise computing has had to govern something it did not build the tooling for in advance. Networks produced authentication and directory services. Distributed applications produced API gateways. Cloud produced container orchestration and secrets management. In each case, the enterprises that moved early to standardized agentic operations captured the structural advantage.

The maturity path for the agent era — Experimental, Adopting, Standardizing — is the same arc compressed into a shorter window. The Experimental stage is necessary; learning happens there. The Adopting stage is unavoidable; the first generation of controls comes from it. The Standardizing stage is the destination, and the only stage that produces operations the estate can run on for the long term.

The question for executive sponsors is not whether to standardize. It is when, and at what cost of waiting. Every quarter spent in Adopting accumulates fragmentation. Every quarter spent in Standardizing compounds returns. The enterprises that move into standardized agentic operations early — while the estate is in the dozens of agents rather than the hundreds — will set the operational baseline the rest of the industry catches up to.

The pattern is recognizable. The platform exists. The choice is the timing.

Read more on the website: the platform, the solutions, or the perspectives series — Another Governance Layer, Again, The Scarcity Isn't Code Anymore, and The New Shadow IT.

Let us know how we can help: hello@domesystems.ai