Promethium

From Pilot to Production:

The operator's playbook for agentic analytics

The CDO walks into the quarterly business review. Six months ago she told the board that the pilot proved it: an AI agent could answer natural-language questions across the data estate, and the demo was good enough to show the CEO. The board is ready for an update.

The honest answer is that the pilot still works on the fifteen questions it was built for, and nothing else. A second domain was supposed to go live last month. The team is quietly rewriting the context layer for the third time. Finance has opinions. Security has questions. Legal has concerns. Nobody outside the pilot team is actually using the thing. No one on the board wants to hear that, so the CDO talks about velocity and momentum instead.

Every operator reading this has been in some version of that room. Six months of spend, a team that is tired, and a board that is losing interest.

The gap between "the demo worked" and "real people use this every day" is where most agentic analytics programs die. The failure point is consistent: the operating system around the model — data access, context, trust, governance — wasn't built to scale.

Some organizations are closing the gap. Cisco's 2025 AI Readiness Index classifies 13% of organizations as Pacesetters. Pacesetters are four times more likely to move AI pilots into production than the average organization, and 62% of them have mature, repeatable pilot-to-production processes. The gap is real, and it is crossable.

This playbook is the practitioner's guide to crossing it. What follows is a set of guides, frameworks, and checklists the operator can work through with their team: an introduction to the five most common failure modes of production agentic analytics, a context readiness audit, a matrix for choosing the right first domain, a reference architecture, an operating model and RACI, a short list of production health metrics, and a launch-ready checklist. Time to ship.

1. The gap that kills agentic analytics programs

The pilot worked. So why isn't it in production?

Plenty of organizations have agents. Very few have agents doing analytics.

BARC's Trend Monitor 2026 puts the gap in sharp relief: half of organizations have AI agents in production, but only 27% use them for analytics specifically. The demo-to-production gap shows up most often in analytics workloads — the ones with the messiest data, the most contested definitions, and the least-forgiving users.

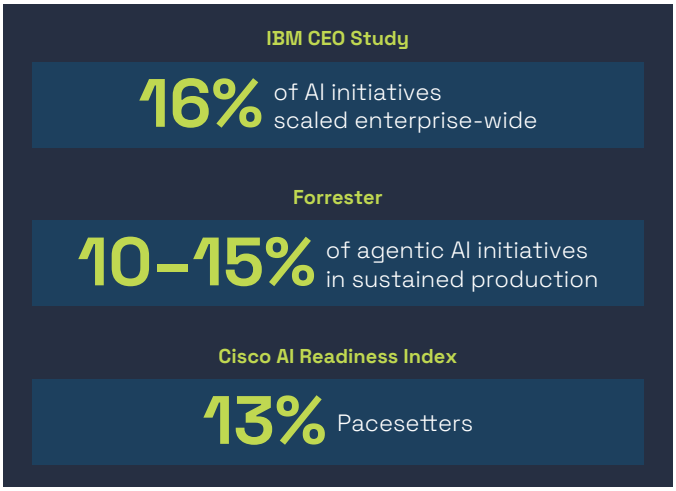

FIGURE 1: The pilot-to-production gap.

The numbers converge. IBM's 2025 CEO Study found only 16% of AI initiatives have scaled enterprise-wide, and only 25% have delivered the ROI CEOs expected. Forrester's Biswajeet Mahapatra put a sharper number on it in January 2026: only 10 to 15 percent of agentic AI initiatives reach sustained production use. Cisco's Pacesetter research lands at 13%. Three methodologies, three independent samples, one conclusion — roughly one in seven programs makes it.

That's the bottleneck the rest of this playbook addresses.

The pilot fools you

Pilots systematically understate how hard production is. The question set is bounded. The data is curated. The users are experts who can forgive a weird answer. The domain is narrow by design. Every one of those assumptions breaks when a broader audience shows up with less-scripted questions.

The practitioner record is full of organizations that burned a million dollars and six months on pilots that never got past the first domain. The root cause is almost always the same: context was hand-engineered for the pilot's question set, so the agent couldn't generalize to the next use case.

Gartner's Anushree Verma named the pattern in a June 2025 press release: "Most agentic AI projects right now are early stage experiments or proof of concepts that are mostly driven by hype and are often misapplied. This can blind organizations to the real cost and complexity of deploying AI agents at scale, stalling projects from moving into production." On the back of that finding, Gartner forecast that more than 40% of agentic AI projects will be canceled by the end of 2027 — the cited drivers being escalating costs, unclear business value, and inadequate risk controls.

Three gaps that compound at scale

Three gaps kill production deployments, and each one is survivable on its own. Together they compound.

The first is data access. The pilot worked because the team put its data in one place. Production asks questions whose answers live across warehouses, lakes, SaaS apps, and operational systems. Most stacks were built for pipelines, not for the live federation an agent needs.

The second is context. The pilot had a handcrafted definition of "revenue" for the one domain that needed it. Production requires multi-dimensional context engineering — structured representations of metrics, rules, hierarchies, and domain-specific nuance that the agent can reason over. Gartner reported at the 2026 D&A Summit that 63% of organizations either lack or are unsure about the data management practices AI requires, and only 17% have reached high AI maturity.

The third is trust. The pilot had expert users who could sanity-check outputs. Production has casual users who trust or abandon the tool based on the first wrong answer — and the first wrong answer is almost never hypothetical. Moffatt v. Air Canada (BC Civil Resolution Tribunal, February 2024) is already the textbook case: a customer-service chatbot invented a bereavement fare refund policy, and the tribunal held the airline liable for what the agent told the

customer. The damages are small. The precedent is what operators and their legal teams are paying attention to.

What production actually means

Most teams overuse the word. A pilot turns into a production deployment when it's used continuously by an audience broader than the original team that built it. Production doesn't require thousands of users or multiple domains — plenty of valuable production deployments are single-domain and serve a few hundred people — but it does require that the audience has expanded past the pilot's controlled perimeter and that usage is sustained, not episodic.

By that bar, widely deployed enterprise copilots often aren't in production either. Practitioner analyses through early 2026 put seat utilization at roughly 10%, meaning nine out of ten licensed users don't open the tool in a given month. Gartner reports only 5% of copilot pilots have moved to larger deployments. License counts flatter the CDO's story in the short term. Usage is the number that matters.

The next chapter starts where most teams' debugging starts — with the model — and explains why the model is almost never the variable worth tuning.

2. Why the same model that nailed the pilot fails in production

Architecture is the variable

Most teams respond to a pilot that breaks in production by upgrading the model. A year of frontier-model upgrades later, they have the same problem.

The cleanest evidence shows up in the academic benchmarks. Spider 1.0 — the long-running text-to-SQL benchmark — runs on clean, academic schemas. Spider 2.0 (ICLR 2025) reuses the same task on enterprise-style schemas averaging 812 columns. The same frontier models that nearly max out on Spider 1.0 collapse on Spider 2.0, dropping from the high eighties to the low single digits. Same models, different context. The accuracy gap that opens between the two benchmarks is roughly the gap between a clean pilot and an honest production deployment.

The diagnosis is structural. Capability is fine. The architecture around the model is the limit, and that is where the rest of this playbook focuses.

The three compounding gaps

Three architectural gaps separate a working pilot from a working production deployment. Each is survivable on its own. Together, they compound.

Data access. Pilots work on curated, local data. Production needs federated, live access across every warehouse, lake, SaaS app, and operational system where the answer might live. Pipeline-based ETL exposes only a fraction of enterprise data to an agent at any given moment; the rest sits in sources the agent can't reach without a copy step. Forrester's Wave for Data Fabric Platforms (Q4 2025) treats federation as required substrate for agentic AI. The pilot worked because its data happened to be in scope. Production breaks the moment a user asks a question whose answer lives in a source the pipelines never reached.

Context. Pilots hand-engineer context for a narrow question set. Production needs multi-dimensional context engineering — the structured, compounding representation of metrics, rules, hierarchies, and domain knowledge that Chapter 4 walks in detail. Hand-curation doesn't survive past the first domain. The teams that scale put a graph underneath the context work; the teams that stall are still copy-pasting metric definitions per use case.

Trust. Pilots have expert users who forgive a weird answer. Production has casual users who abandon the tool on the first wrong one — and when the agent is customer-facing, "weird" becomes "legally actionable" quickly. Air Canada paid small damages in the Moffatt case (BC Civil Resolution Tribunal, February 2024), but the precedent — that enterprises own their agents' output — is what operators will be citing for the next decade.

The next chapter names the five failure modes that show up when any one of these gaps goes unaddressed, and the rest of the playbook is how the operator closes them.

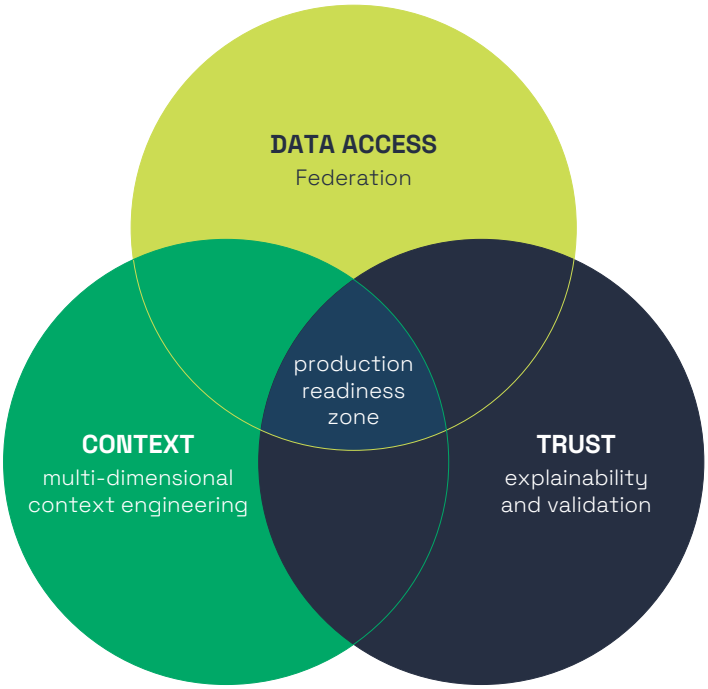

FIGURE 2: The three compounding gaps.

3. The five failure modes of production agentic analytics

Why there are patterns

Pilots fail in the same ways over and over. Every operator who has done this more than once has seen most of these before, often at the same time. Naming the patterns is the first step in choosing which one to fix first and in making sure the rest of the team is debugging the same problem.

The five failure modes below are drawn from the pattern that repeats across the research (BARC, Gartner, Forrester, Cisco, Gravitee, CSA/Zenity) and from the practitioner record. Each one has a recognizable symptom, a root cause that lives in the architecture rather than the model, and a fix that shows up as one of the three reference-architecture layers in Chapter 6.

A short self-check appears at the end of the chapter. Read the modes first, then use the checklist to flag which ones are showing up in the deployment today.

Failure mode 1. Context collapse

Symptom. The agent is confident and accurate on the pilot question set and falls apart outside it. Questions that use a slightly different phrasing, cross two domains, or depend on an unwritten business rule return wrong answers, refusals, or plausible-looking hallucinations.

Root cause. Context was hand-engineered for the pilot. The implicit semantic model sits in code, prompts, or one-off configuration files. It can't generalize because it was never structured to.

Pattern in the research. BARC's Trend Monitor 2026 found knowledge graph adoption in production at 27% in late 2025, up from 26% in early 2024. Context infrastructure is stagnating while agent deployment accelerates. Teams that skip multi-dimensional context engineering are hoping the model closes the gap, and Chapter 2's benchmark numbers show that it doesn't.

Fix. Multi-dimensional context engineering structured in a graph that every domain rollout enriches. Not hand-curation per domain. The layers and the audit are in Chapter 4; the architectural home is Layer 2 in Chapter 6.

Failure mode 2. Data access dead ends

Symptom. The agent answers confidently when the answer lives inside the warehouse, and falls back to "I don't have that data" when it lives anywhere else. Users learn quickly which questions work and stop asking the others.

Root cause. Pipeline-based architectures assume data will be copied into a central store. Agents need live, federated access. The two architectures optimize for different things.

Pattern in the research. Industry analysis consistently shows that pipeline-based ETL exposes only a minority of enterprise data to agents at any moment, against roughly universal coverage for federated zero-copy architectures. The pilot worked because the pipeline happened to cover the data the pilot asked about. Production breaks as soon as users ask questions whose answers live in the sources the pipelines never reached.

Fix. Federated, zero-copy query execution as the bottom layer of the reference architecture (Layer 1 in Chapter 6). No new ETL unless a specific use case requires it.

Failure mode 3. Trust erosion

Symptom. Adoption grows during the pilot, peaks early, and then declines. Power users who championed the tool stop using it. The team keeps shipping improvements and the numbers keep dropping.

Root cause. The first wrong answer in front of a new user kills the second chance. The second kills the relationship. Without explainability — the SQL behind the answer, the lineage, the rules applied — users can't tell a correct answer from a confident wrong one.

Pattern in the research. Moffatt v. Air Canada is the legal anchor: the Civil Resolution Tribunal's February 2024 ruling established that enterprises own their agents' outputs. The scale version of the same pattern shows up in copilot adoption data, where widely deployed copilot suites sit at roughly 10% seat utilization through early 2026 and Gartner reports only 5% of copilot pilots scale. The root cause is almost always the same — inconsistent answer quality eroded user trust before critical mass formed.

Fix. Explainability and validation at query time. Every answer carries its SQL, its lineage, and the rules that governed it. The architectural home is Layer 3 in Chapter 6.

Failure mode 4. Governance collision

Symptom. The agent either refuses to answer questions it should answer (because permissions were modeled incorrectly), or answers questions it shouldn't (because it inherited a human's broader access). Security has filed tickets. Legal has concerns. Audit is uncomfortable.

Root cause. Governance was retrofit after the agent was built. Row-level, column-level, and policy-based access controls exist in the warehouse but don't flow through the agent's query path. Most deployments hand agents shared API keys or inherited user permissions — neither of which the governance stack was designed to enforce.

Pattern in the research. Gravitee's 2026 State of AI Agent Security survey found that 88% of organizations had confirmed or suspected AI agent security incidents in the past year. The CSA and Zenity 2026 survey went one level deeper: 53% of organizations reported agents exceeding intended permissions occasionally or more often, and only 18% base AI agent access on the agent's own permissions. The rest inherit human-user permissions or shared API keys, which means structurally the agent has more access than any one user it serves.

Fix. Governance enforced at query time, against the agent's own identity, with row/column/policy rules applied at the semantic layer rather than as a post-hoc filter. Layer 3 again.

Failure mode 5. Maintenance tax

Symptom. Every new domain takes as long as the last one. The team never reaches a rhythm. The second domain should be faster than the first, and the third faster still, but every new rollout reopens the same context-engineering work.

Root cause. Context artifacts are being rebuilt per domain instead of extended from a shared graph. The platform has no compounding asset.

Pattern in the research. Gartner's 2026 D&A Summit reported that 63% of organizations lack AI-ready data management practices, and BARC found knowledge graph adoption flat year-over-year despite rising agent deployment. Hand-curated semantic models and column-description glossaries don't survive the combinatorial load of domains, users, and question variety.

Fix. A context graph that gets richer with every correction, every endorsement, and every new domain. The work done in domain one becomes input to domain two.

Failure mode	Symptom	Root cause	Fix
Context collapse			
Data access dead ends			
Trust Erosion			
Governance collision			
Maintenance tax			

FIGURE 3: The five failure modes, at a glance.

Reading the five modes

The five modes are not independent. Context collapse and maintenance tax are usually two symptoms of the same missing graph. Data access dead ends and governance collision frequently co-occur, because the absence of a federated query layer means governance gets applied in too many places to enforce consistently.

Quick self-check

Use this self-check with the team to flag which modes are showing up in the deployment today. One question per mode — answer yes or no.

- ☐ **Context collapse.** Does the agent fall apart outside the pilot's question set, or does every new domain reopen the context work from scratch?
- ☐ **Data access dead ends.** Do users regularly ask questions whose answers live in a source the agent can't reach without a pipeline request?
- ☐ **Trust erosion.** Can users *not* see the SQL, lineage, or rules behind an answer — and is adoption flat or declining?
- ☐ **Governance collision.** Does the agent run on a shared API key or human-inherited permissions rather than its own identity, and can the audit trail not fully reconstruct what the agent did?
- ☐ **Maintenance tax.** Is the second domain taking as long as — or longer than — the first?

Two or more "yes" answers means the deployment isn't ready for a broader rollout; work the relevant chapters (4 through 9) before planning a launch. One "yes" tells the team where to focus first.

With the failure modes named, the next chapter goes deep on the one that does the most damage — context — and provides an audit the operator can run inside a week.

4. The context readiness audit

What this chapter is

Chapter 2 made the case that multi-dimensional context engineering is the single biggest accuracy lever an operator controls. This chapter is the audit for it. The first half introduces the five dimensions of context that need to be in place. The second half is a working checklist the operator can run against each in-scope domain to produce a score, a stoplight, and a list of thirty-day quick wins.

Why context readiness matters before you ship

Most pilot-to-production failures trace to context gaps the team didn't know it had. The team knows the model. The team knows the warehouse. The team doesn't always know that its "enterprise account" definition lives in four different BI tools and three different slide decks, with three different thresholds, and that none of them are codified anywhere the agent can see.

Salesforce's Muralidhar Krishnaprasad put the argument in one sentence in Salesforce Newsroom in January 2026: "this is why context is really important for an agent: for it to solve the problem, solve it the right way, and also for us to make sure that it's giving you the right ROI." Forrester Consulting's April 2026 survey (commissioned by Simpplr) reinforced the point — 45% of IT leaders say missing organizational context is the main reason AI fails to deliver expected results, and 85% say fragmented data and knowledge systems must be unified for AI to succeed.

Gartner's official March 2026 definition of context engineering, quoted in Chapter 2, is operational. The audit below turns it into something a team can score.

The five dimensions of enterprise context

Every production deployment needs all five. Most have one or two. This is what "multi-dimensional context engineering" actually means in practice.

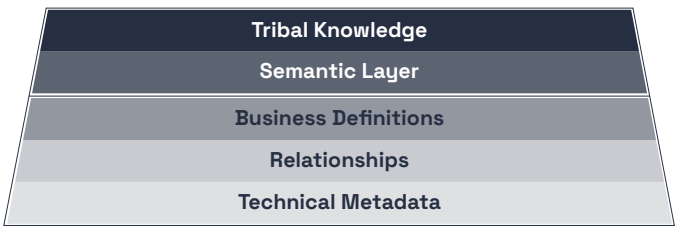

FIGURE 4: The five dimensions of multi-dimensional context engineering.

Technical metadata. Schemas, column types, primary keys, joins, data types. Every catalog has this. Necessary, not sufficient.

Relationships. Foreign keys, join paths, and the links between entities across data sources. The agent needs to know that "customer" in Salesforce is the same customer in the warehouse, and which keys connect them. Often implicit in the data, rarely explicit anywhere.

Catalog and business definitions. What does "revenue" mean in this company? "Customer"? "Active user"? These usually live in a data catalog, a glossary, or tribal knowledge — and often conflict across departments. The dictionary layer.

Semantic layer. Metric definitions and calculation logic — how revenue is computed, how churn is calculated, the thresholds and hierarchies that frame both. The "rules and math" layer that typically lives in BI tools, dbt models, or scattered SQL.

Tribal knowledge and memory. Domain expertise that lives in people's heads, decisions made for reasons that never got documented, and the user-specific memory that personalizes results to a role and a query history. The hardest layer to capture, and the one that makes the agent feel like it understands the business at scale.

The audit — a scoreable checklist

Use the checklist below with the Context lead and the Domain lead for the domain being assessed. For each item, score ☐ (not in place), ◐ (partial), or ☑ (in place). Total the check marks across all five layers to produce a score out of 25.

Checklist: technical metadata

- ☐ Schemas, column types, and primary keys documented for every in-scope table
- ☐ Single source of truth for table and column documentation (no duplicated glossaries)
- ☐ Data freshness and refresh cadence documented at the table level
- ☐ Automated metadata capture in place, not maintained in a spreadsheet
- ☐ Metadata refreshed as schemas change rather than in periodic catch-up sweeps

Checklist: relationships

- ☐ Foreign keys and join paths made explicit, not buried in query code
- ☐ Cross-source relationships mapped (e.g., Salesforce customer ↔ warehouse customer)
- ☐ Join logic version-controlled and reusable, not rewritten per use case
- ☐ Relationships represented in a queryable form (graph, semantic layer), not only diagrams
- ☐ New relationships added through a defined process as new sources come online

Checklist: catalog and business definitions

- ☐ A canonical definition for every top business metric and entity
- ☐ Definitions are machine-readable, not only prose
- ☐ Synonyms and aliases captured (e.g., "ARR" resolves to "Annual Recurring Revenue")
- ☐ Cross-departmental conflicts explicitly reconciled and documented
- ☐ Business-owner sign-off on the definition list, with version control

Checklist: semantic layer

- ☐ Metrics defined once and reused (no parallel definitions in BI tools)
- ☐ Calculation logic version-controlled and auditable
- ☐ Thresholds ("enterprise," "churn," "high-value") codified centrally
- ☐ Hierarchies (org, geography, product) represented as structured data
- ☐ A change-management process for metric and rule updates

Checklist: tribal knowledge and memory

- ☐ Decision traces and "why we do it this way" notes captured for top metrics
- ☐ Domain experts have a structured way to add context as it surfaces in use
- ☐ User corrections, endorsements, and flags feed back into the context layer
- ☐ Personalization based on user role and query history is wired in
- ☐ Reinforcement loop is running — corrections change future answers, not only dashboards

Interpreting the score

- **Under 10 — red.** The deployment can't safely expand beyond the original pilot scope until context is built out. Prioritize technical metadata, relationships, and catalog/business definitions for the next sprint.
- **10 to 18 — yellow.** The deployment can support a limited production rollout in the strongest domain while context work continues in parallel.
- **19 or higher — green.** The context base is strong enough to support production at the in-scope domain and to start compounding through the reinforcement loop.

Most organizations score yellow on their strongest domain and red on the rest. That's normal. It's also why domain selection (Chapter 5) matters more than team velocity.

Quick wins by layer

Three concrete actions per dimension that an operator can take in the first thirty days:

- **Technical metadata** — automate metadata capture, retire duplicate catalogs, fix the top-50 undocumented columns.
- **Relationships** — make the top cross-source joins explicit, get the join logic out of query code into a queryable form.
- **Catalog and business definitions** — codify the top business metrics, version-control them, collect the business-owner sign-off list.
- **Semantic layer** — pull metric calculations out of BI tools into a single semantic definition, version-control the rules, publish a change log.
- **Tribal knowledge and memory** — wire the reinforcement loop so every correction updates the context graph, and stand up a structured way for domain experts to add context as it surfaces.

Green on technical metadata, business semantics, and decision logic, with yellow on the last two dimensions, is enough to ship. The last two mature as the graph compounds with use. Waiting for green everywhere before launch is usually a sign of political discomfort with domain ownership rather than a real readiness gap.

5. Domain selection: don't start where you think

The most common mistake

The CDO has promised the board that AI will transform analytics, so the team picks the most visible domain — usually finance or executive reporting — as the first production deployment.

That's the wrong call.

High visibility is also high scrutiny and high political cost on failure. The first production deployment is where the architecture gets shaken out, and it will have rough edges. A CFO who sees an agent return two different numbers for Q2 revenue remembers it for a year.

Context readiness. Use the Chapter 4 audit score for each candidate domain. Most teams have done technical-metadata and business-semantics work in one or two domains already — those are the readiness-green candidates. The rest are further behind.

Value concentration. A domain has concentrated value when a small set of decisions drives measurable outcomes. Operations analytics (supply chain exceptions, customer service call drivers) and product analytics (activation funnel, feature adoption) usually have concentrated value. Executive reporting usually has diffuse value — many people consume many dashboards, but few decisions flow directly from them.

FIGURE 5: The domain selection matrix.

The matrix produces four quadrants:

- **Start here — high readiness, concentrated value.** Usually operations or product analytics. Clear wins, contained failure domain, users who can validate answers.
- **Plan for second — high readiness, diffuse value.** Revenue operations, marketing analytics. Good candidates once the flywheel is turning.
- **Earn the right — low readiness, concentrated value.** Finance, compliance analytics. Build the context before the agent.
- **Avoid for now — low readiness, diffuse value.** Often executive reporting. The political prize and the hardest place to land the first launch.

Why domain two matters more than domain one

Domain one proves the pattern. Domain two is where the architecture earns its keep. If the second domain takes as long as the first, the context graph isn't compounding and the team is back on the maintenance-tax treadmill.

A realistic benchmark: domain two should take roughly half the time of domain one. By domain four, subsequent rollouts should be measured in weeks. The acceleration comes from the graph carrying more of the work each time, not from a team that got faster. The team is usually the same size or smaller than it was during the pilot.

The four-week-per-domain target

Four weeks per domain is the Pacesetter tempo, not the industry average. ModelOp and Corinium's 2025 AI Governance Benchmark Report found that 56% of generative AI initiatives take six to eighteen months to move from intake to production. The four-week figure is aggressive on purpose — it's the tempo that makes the ROI math work and the tempo the reference architecture is designed to support.

Teams hit it through three things: a compounding context graph, a reinforcement loop that captures domain knowledge as a byproduct of use, and a shared playbook (this one) that stops the team from re-deciding the same architecture questions every time a new domain comes online.

6. The architectural prerequisites for production

The three architectural prerequisites

Every production-grade agentic analytics stack has three architectural layers. Teams can build them, assemble them from best-of-breed components, or buy them as a single platform. Skipping any of them lands the program back in the 84% of deployments that never reach production.

1. **A federated query layer.** Live, zero-copy access across every data source where an answer might live.
2. **A context graph layer.** A compounding artifact that carries multi-dimensional context across domains.
3. **A trust and governance layer.** Explainability, validation, and policy enforcement at query time.

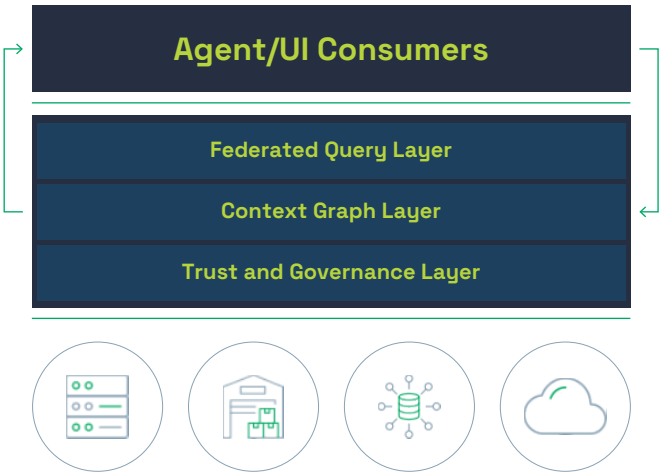

FIGURE 6: The three-layer reference architecture.

Each layer addresses a specific failure mode from Chapter 3. The sections below lay out what each layer does and why it can't be skipped.

Layer 1. The federated query layer

The federated query layer lets an agent reach any data it needs without the enterprise copying the data first.

The case for federation comes straight out of the data access gap in Chapter 3. Industry analysis puts pipeline-based ETL coverage at a minority of enterprise data at any moment, against roughly universal coverage for federated zero-copy architectures. Forrester's Wave for Data Fabric Platforms (Q4 2025) treats federation as required substrate for agentic AI.

Federation at the query layer does three things pipelines can't:

- It reaches data wherever it lives — warehouse, lake, SaaS app, operational database — without waiting for pipelines to catch up.
- It stays live, so agents answer questions against current data rather than last night's snapshot.
- It isolates governance at the query point, so one policy layer applies to all access.

Moving from an ETL-first stack to a federation-first stack is an architecture simplification. The team maintains fewer pipelines, governs fewer copies of the same data, and stops waiting on the nightly batch to catch up to the question.

Layer 2. The context graph layer

The context graph layer is the compounding asset of the architecture. It carries the five dimensions of multi-dimensional context engineering (Chapter 4) across every domain, every user, every correction.

Treat catalogs and semantic layers as inputs that feed the context graph — they don't replace it. A catalog gives the agent the technical-metadata dimension. A semantic layer gives the agent some of the business-semantics dimension. Neither captures decision logic, domain rules, or experiential context in a form an agent can reason over across domains.

The graph compounds because every interaction feeds back into it. A correction from a business user updates the relevant node. An endorsement raises the confidence weight on a specific answer path. A new domain extends the graph with entities and relationships that become input to every other domain's queries. Domain three doesn't start from zero; it starts from the work domain one and domain two already did.

Gartner's research prediction quantifies the payoff: by 2028, context engineering features will be built into 80% of software tools used to build AI applications, and context engineering improvements will enhance agentic AI accuracy by at least 30%. Salesforce's

Rahul Auradkar framed it more pointedly in January 2026: "in the agentic enterprise, where humans and agents work together, context has become the new currency."

Layer 3. The trust and governance layer

The trust and governance layer does three jobs:

- **Explainability.** Every answer carries its SQL, its lineage, and the rules applied. A user who questions an answer can see the derivation. An auditor can trace the policy path. A developer can debug the failure mode without rerunning the query.
- **Validation.** Answers are checked against a regression set before they are shown. Accuracy on the regression set is a gating criterion for every release.
- **Policy enforcement.** Row-level, column-level, and policy-based access controls are applied at the query layer against the agent's own identity — not inherited from a human user or a shared API key.

Governance bolted on after the fact doesn't hold at production scale. Gravitee's 2026 survey found 88% of organizations had confirmed or suspected AI agent security incidents in the past year; CSA and Zenity found that only 18% of organizations base AI agent access on the agent's own identity. Layer 3 has to be designed into the architecture.

Build, assemble, or buy

Three honest paths:

- **Build.** Appropriate when the team has a long-term AI platform mandate, staffing for a multi-year investment, and a use-case surface that justifies the effort. Longest time to production and highest total cost, but produces a platform that exactly matches the organization.
- **Assemble best-of-breed.** Pick a federation product, a context graph product, and a governance stack; integrate them. Works when the team has strong integration capacity. The integration tax is the hidden cost — each layer has to stay compatible as vendors ship independently.
- **Buy a platform.** A single vendor implementing all three layers. Fastest time to production; the trade-off is vendor alignment, because the platform's choices about how the layers connect become the organization's choices.

The right answer depends on team size, time pressure, and long-term AI strategy. The wrong answer is to pick a path without a clear three-layer mental model — a team buying or building components without the architecture ends up with two layers and a deployment that doesn't scale.

Promethium's Mantra AI Insights Fabric

Mantra is one implementation of the three-layer architecture, delivered as a single platform. It is relevant to operators for three reasons that map directly back to the failure modes this playbook diagnoses.

First, Mantra's federated query execution addresses the data-access dead ends from Chapter 3. The platform queries data in place across warehouses, lakes, SaaS apps, and operational systems. The team doesn't build pipelines to get the agent to the data, and no additional copy of the data is created for the agent to reason over. Chapter 2's "only a minority of enterprise data is reachable through pipelines" problem disappears on day one.

Second, Mantra's Insights Context Graph addresses the context collapse and maintenance tax modes. The graph holds the five dimensions of multi-dimensional context engineering in a machine-readable form, and it compounds with every correction and every new domain. Teams that have assembled this capability themselves — catalog plus semantic layer plus a home-built glossary plus a brittle prompt orchestration layer — know what that integration tax looks like. Mantra ships the compounding asset rather than asking the team to build it.

Third, Mantra's trust layer addresses the trust erosion and governance collision modes. Every answer carries its SQL, lineage, and the rules applied. Row-level, column-level, and policy-based enforcement happens at the query layer against the agent's own identity. That changes the numbers behind the 88% agent-incident rate for organizations running Mantra.

The bigger point: Mantra compresses the context engineering time window. The closing chapter returns to the tempo customers actually see. For teams evaluating vendors, the three-layer reference architecture above is the scorecard. Mantra's claim is that it scores well across all three layers in a single platform.

7. The operating model: who owns what

Four roles

Agentic analytics isn't a pure data-team project. It sits across data engineering, analytics engineering, platform, and the business. Four roles carry the work:

- **Platform lead.** Owns the federated query layer and the runtime. Makes the build-vs-buy call for Layer 1. Responsible for uptime, responsiveness, and integration depth. Reports to the head of data or head of platform.
- **Context lead.** Owns the context graph. Curates the top business metrics in the first quarter. Runs the reinforcement loop. Makes the cross-department reconciliation calls. Reports to the head of data.
- **Domain lead (one per active domain).** Owns the use case. Signs off on domain rules. Manages user rollout and change management inside the business unit. Reports into the business-unit leader with a dotted line to data.
- **Trust lead.** Owns validation, explainability, and policy enforcement. The person who says "not ready to ship" when accuracy drops below the bar. Often the same person who owns data governance already.

In a small organization, one person can carry two roles. In a large organization, each role is a small team. What doesn't work is leaving any role unassigned — the pilot that shipped without a Context lead is why domain two is still in re-engineering six months later.

Forrester's federated DAAI operating model is the closest published reference: a central platform team plus domain-aligned pods. McKinsey's 2025 agentic-organization research reports that two to five humans can supervise an agent factory of fifty to a hundred specialized agents. The pod sizing is deliberately small, and it only works because the platform does most of the heavy lifting.

RACI for the first domain

A practical RACI table across the five phases of domain rollout. Owners are the four roles above.

Phase	Responsible	Accountable	Consulted	Informed
Discovery (use case, value case)	Domain lead	Business-unit leader	Platform lead, Context lead	Trust lead, IT, Security
Context build (5 dimensions)	Context lead	Head of data	Domain lead, Data engineering	Trust lead, Platform lead
Validation (accuracy, policy)	Trust lead	Head of data	Platform lead, Context lead	Business-unit leader, Security, Legal
Rollout (users, enablement)	Domain lead	Business-unit leader	Trust lead, Platform lead	Context lead, Communications
Measurement (health metrics)	Platform lead	Head of data	Trust lead, Domain lead	Business-unit leader, CDO

Teams without a Trust lead usually end up with a CDO doing the Trust lead's job, which works for one domain and fails at the second.

The reinforcement loop

The reinforcement loop is the mechanism by which the context graph gets richer with use. Every user correction, endorsement, and flag is a signal, the signals feed the graph, and the graph changes the next answer.

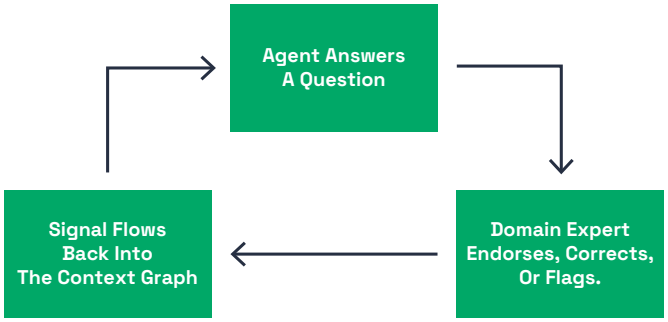

FIGURE 7: The reinforcement loop.

Three concrete requirements for a working loop:

- Corrections from domain experts are first-class events — logged, attributed, and applied — not stray comments in a feedback channel.
- Endorsements (thumbs up on an answer, confirmation from a domain expert) raise the confidence weight on the path that produced the answer.
- Flags (something is wrong but the user can't specify what) trigger a review by the Context lead rather than disappearing.

If the loop isn't running, the platform isn't compounding. Chapter 8's health metrics measure whether it is.

Staffing realistically

The common mistake is front-loading eight to ten FTE during the pilot and having nothing to show for it. Once the reference architecture is in place, most teams run production at two to four dedicated FTE plus shared capacity on platform and data engineering.

McKinsey's 2025 State of AI found that only 5.5% of companies see material financial returns from generative AI, and fewer than 10% are scaling AI agents across multiple functions. The failure pattern is consistent — layering AI on top of workflows never designed for it, instead of redesigning the operating model. The four-role structure above is the minimum redesign.

Working with IT, security, and legal

Production means IT, security, and legal are on the critical path, not in a final review at the end.

- **IT** owns identity and access. The agent needs its own identity so the trust and governance layer can enforce policy. Get this on the IT roadmap at the start, not after validation.
- **Security** owns incident response. Given the 88% agent-incident rate in the Gravitee survey, the security team should be building the incident playbook in parallel with rollout, not afterward.
- **Legal** owns regulatory and contractual risk. The Moffatt v. Air Canada precedent is settled law in Canada; similar cases are moving in other jurisdictions. Legal should approve the explainability bar and any customer-facing disclosure language before the first external-facing deployment.

Gartner's Sarah James made the political backdrop explicit at the Data & Analytics Summit in May 2025: by 2027, 75% of CDAOs not seen as essential to their organization's AI success will lose their C-level position. The operator's work is also the CDAO's career, so the operating model isn't purely an infrastructure exercise.

8. Production health metrics

Five metrics that matter

Most teams measure a dozen things badly and the handful of things that matter not at all. The five metrics below are sufficient to know whether a production deployment is working — whether it serves one domain or ten.

FIGURE 8: Production health dashboard mockup.

1. **Answer accuracy rate.** Percent of answers validated correct by domain experts or against a golden benchmark set. Leading indicator of trust.
   - Target: reach a defined accuracy baseline agreed with the domain owner before expanding the audience.
   - Failure signal: accuracy dropping week-over-week in an already-live domain almost always means context drift — metric definitions changed, a new data source was added, a domain owner left.
2. **Time-to-answer.** Wall-clock time from question submitted to trusted answer displayed. Leading indicator of adoption.
   - Target: fast enough that users don't drop off. Interactive applications lose a significant share of users when responses stretch past a few seconds, so "fast" isn't a nice-to-have — it's an adoption gate.
   - Failure signal: users opening the tool, waiting through a long spinner, and closing it without asking a second question. Session-length drops are the early warning.
3. **Question coverage.** Share of the questions users actually ask that the agent can answer confidently. Leading indicator of context sufficiency.
   - Target: question coverage climbing over the first ninety days of live usage. Coverage is what the graph buys the team — more of the question space gets covered as more context is captured.
   - Failure signal: coverage flat while corrections are piling up means corrections aren't flowing back into the graph. The reinforcement loop is broken.
4. **User trust score.** Ratio of endorsements to corrections on answers, measured through the reinforcement loop rather than through surveys. Lagging indicator of stickiness.
   - Target: a ratio that improves over the first quarter and stabilizes high. An abstention rate (agent declines to answer) in the middle single digits is healthy — very low abstention means the agent is over-confident, very high abstention means coverage gaps.
   - Failure signal: corrections accelerating without a matching rise in endorsements means users are policing the tool rather than trusting it.
5. **Active adoption.** Share of the target audience using the deployment on a sustained cadence (weekly or monthly depending on the use case). Lagging indicator of value delivered.
   - Target: adoption climbing through the first two quarters and stabilizing high within the target audience for the use case, whether that audience is a hundred people or ten thousand.
   - Failure signal: enterprise copilot-style "licensed to fail" patterns — high license count, low active usage. If the target audience isn't using the tool, seat counts in the board deck flatter the program without doing anything to justify it.

Instrumenting each metric

Where the data lives and who reviews it:

- **Accuracy** — instrument at the agent output layer. Domain experts review a stratified sample weekly in the first quarter, monthly thereafter. Golden benchmark runs on every release.

- **Time-to-answer** — instrument at the federated query layer. Platform lead reviews daily during rollout, weekly in steady state.
- **Question coverage** — instrument at the context graph layer. Context lead reviews monthly.
- **Trust score** — instrument in the reinforcement loop. Domain lead reviews weekly with the business-unit leader.
- **Active adoption** — instrument at the consumer interface. Domain lead and business-unit leader review monthly.

Leading versus lagging

Accuracy, time-to-answer, and question coverage move first. Trust score and active adoption lag by thirty to ninety days. Slow movement on lagging indicators isn't cause for concern in the first quarter. Stalled leading indicators are — accuracy flat after a full quarter of context work means the context graph isn't getting richer, which means the reinforcement loop is broken.

The expansion trigger

Green on accuracy, time-to-answer, and question coverage for sixty consecutive days is the go signal for expanding the audience or adding a second domain. Anything earlier is a gamble.

9. The production readiness checklist

How to run this checklist

Run the checklist two weeks before go-live with the Platform lead, Context lead, Domain lead, and Trust lead in the room. Work through the five sections below. Each question is a yes or no, with a space to capture notes, owners, or open items. A "no" isn't a blocker on its own — it's a prompt to decide whether to resolve before launch, document as a known gap, or accept and move forward.

Data access

☐ **Do we know where all the data this use case depends on lives?**
List the warehouses, lakes, SaaS apps, and operational systems the agent needs to reach. Missing sources are the most common launch surprise.
Notes: _____

☐ **Do we have access to all of it — permissions, credentials, and connectivity in place end to end?**
Access that works in isolation doesn't always work when every source has to be queried together. Test the full path.
Notes: _____

☐ **Can we query every source in one place, at the same time, without copying it first?**
Federated, zero-copy access is the bar. Pipeline-based ETL becomes a bottleneck the moment the second domain goes live.
Notes: _____

☐ **Does the data meet the freshness and quality SLAs the use case requires?**
Stale data and quality gaps silently erode trust. Define the SLA explicitly and confirm it holds for every source in scope.
Notes: _____

☐ **Can we monitor cost effectively as query volume grows?**
Agent-generated query volume is typically an order of magnitude higher than traditional BI. Per-query and per-user cost visibility is the control plane.
Notes: _____

Context

☐ **Are the core pieces of context defined across the relevant dimensions (technical metadata, business semantics, decision logic, domain rules, experiential context)?**
Use the Chapter 4 audit. Green on at least the first three dimensions is the minimum for launch.
Notes: _____

☐ **Is all the necessary context accessible to the agent at query time?**
Context that lives in a Confluence page or a BI glossary the agent can't read doesn't count.
Notes: _____

☐ **Have SMEs and domain owners reviewed and signed off on the existing context — and is there a mechanism for them to flag and add missing context as it surfaces?**
Sign-off establishes the launch baseline. The feedback mechanism is what keeps the context graph compounding after go-live.
Notes: _____

☐ **Has the accuracy baseline been agreed with the domain owner and reached on the golden question set?**
The domain owner — not the data team — sets the bar. The golden question set is the measurable gate.
Notes: _____

☐ **Are results personalized based on the user's role and query history?**
Personalization reduces answer variance between users and makes the agent feel like it knows the context each user works in.
Notes: _____

Trust and validation

☐ **Are answers explainable — SQL, lineage, a natural-language explanation, and the rules applied?**
All four elements. Any one on its own is insufficient for a business user to build trust.
Notes: _____

☐ **Are results validated before being surfaced (LLM-as-judge checks, SQL syntactically correct and executable, and similar guardrails)?**
Pre-surface validation catches most confident-wrong answers before a user ever sees them.
Notes: _____

☐ **Are users presented with a trust or confidence score that reflects the context available for the question?**
A score makes the agent honest about when it's guessing. Users calibrate their trust accordingly.
Notes: _____

☐ **Can users endorse correct outputs and flag wrong ones, and do those signals feed the context graph?**
This is the reinforcement loop. Without it, the platform stops compounding.
Notes: _____

☐ **How does the agent handle ambiguity — does it ask a clarifying question rather than guessing?** Abstention and clarification are features, not failures. Tune the thresholds explicitly.
Notes: _____

Governance

☐ **Is fine-grained access control in place at the row, column, and policy level?** Warehouse-level entitlements are not enough. Enforcement has to be field-aware.
Notes: _____

☐ **Is the agent operating under its own identity rather than a shared API key or human-inherited permissions?** Agent identity is what lets the governance stack distinguish agent actions from user actions.
Notes: _____

☐ **Are audit trails capturing every agent action, not only user-initiated requests?** Agent-to-agent calls and autonomous retries have to be in the log too.
Notes: _____

☐ **Are rules for obfuscation and masking defined and enforced?** PII, financial data, and regulated fields need masking rules applied at the query layer.
Notes: _____

☐ **Are policies enforced at query time and sensitive to which user is asking?** Access-time enforcement alone isn't enough — the agent's effective permissions change per query based on the user context.
Notes: _____

Readiness and change management

☐ **Are the four roles assigned (Platform lead, Context lead, Domain lead, Trust lead)?** Any unassigned role is a single point of failure. Small teams can double up, but no role should be empty.
Notes: _____

☐ **Is user-facing documentation in place?** Users need to know what the agent can and cannot do. Set expectations explicitly.
Notes: _____

☐ **Has user training been delivered?** Training moves adoption from curious to sustained. Budget for a second session ninety days post-launch.
Notes: _____

☐ **Is health-metric tracking (Chapter 8) live and visible to the operating team?** The metrics are only useful if they're reviewed on the cadence Chapter 8 specifies.
Notes: _____

☐ **Is the rollout plan documented, with a rollback path if a release breaks production?** Expansion sequence, communications plan, and revert procedure all documented before go-live.
Notes: _____

Using the results

Tally the “yes” answers within each of the five sections separately. The section with the lowest score is the weakest pillar of the deployment — and the one the program is most likely to trip on in production. Focus attention there before expanding scope or adding a second domain. A team that is green on data access and trust but weak on context and governance will hit a different set of problems than a team that is green on context but weak on data access. The shape of the gap matters, not just the total count.

A clean pass — “yes” across every question — is rare and usually a sign the team is being generous with itself. What matters is that every “no” is either closed before launch, documented as a known gap the team will address in the first ninety days, or explicitly accepted at the leadership level. Delay should be a function of evidence rather than politics. A CDO who brings this checklist with honest answers to the executive team knows exactly why launch is slipping and can defend it to the board.

10. Compressing the context engineering time window

Agentic analytics is a non-trivial problem for a team to solve, and it becomes considerably harder once the program tries to scale beyond the initial pilot and stand the capability up company-wide. The pattern most operators see is that the work stays linear across domains — each new domain reopens the same context engineering, the same metric arguments, the same governance negotiations, the same trust-building cycle that domain one already went through. Work that stays linear, rather than compounding, puts the ROI math under pressure.

The honest industry reality is that most teams need six to twelve months to get the first domain into production, and then roughly the same window again to bring the second domain live. By the time domain three is on the table, budget and patience have usually run out. R Systems' Neeraj Abhyankar put the timeline pressure directly in CIO.com in October 2025: "in the next 12 to 18 months, context engineering will move from being an innovation differentiator to a foundational element of enterprise AI infrastructure." Teams on the linear treadmill are falling behind that window, not closing it.

Building a production-grade agentic analytics stack in-house is tough — and the difficulty grows sharply at scale, where the context graph has to compound across domains, the validation loop has to stay reliable under load, and the governance surface has to keep pace with a rising number of users and use cases. Most teams that try it end up maintaining three or four loosely integrated components and paying a significant integration tax every time one of them ships a breaking change.

What customers see on Mantra

Promethium customers compress the timeline significantly.

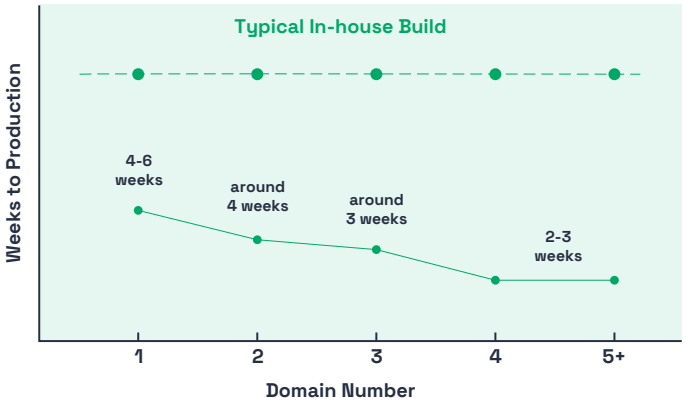

FIGURE 9: Time-to-domain curve with Promethium.

Three things make the compression possible.

- **Targeted first domains, with existing context investments reused.** The first domain goes live in four to six weeks when the team picks the use case carefully (Chapter 5) and feeds existing context investments — catalogs, semantic layers, BI definitions, tribal knowledge — into the Insights Context Graph inside Promethium's Context Hub. The graph ingests what the team already has, rather than asking the team to rebuild it from scratch.
- **A robust validation and reinforcement layer that SMEs can run themselves.** Results are validated before being surfaced, and SMEs can endorse correct answers, flag wrong ones, and add missing context directly into the graph. The data team no longer has to model every definition perfectly upfront — the context graph gets richer with each correction.
- **A graph that compounds across domains.** Domain two typically lands around four weeks, domain three around three, and the fourth and beyond in the two-to-three-week range. The work done for earlier domains carries into the next one, so the team spends progressively less time rebuilding context and more time expanding the surface area of the deployment.

That tempo is what turns the agentic analytics ROI math from unworkable into compelling. Promethium offers a faster path to production-grade accuracy at scale. Reach out to learn how Mantra can help in your ecosystem.

Promethium

About Promethium

Promethium builds Mantra, the AI Insights Fabric that lets enterprise data teams move agentic analytics from pilot to production in weeks rather than months. Mantra combines federated data access, a compounding Insights Context Graph, and trust and governance at query time — the three architectural layers every production-grade agentic analytics deployment needs — in a single platform. Customers across financial services, healthcare, retail, and technology use Mantra to scale agentic analytics across domains without rebuilding context for each one.

Learn more at promethium.ai