ENTERPRISE AI PLAYBOOK · DEEP DIVE · PART 3 · BUILD

Operationalizing LLMs as Enterprise Tools

A six-layer operating model for taking large language models from promising pilot to governed, production-grade enterprise capability.

THE BOTTOM LINE

Gartner found that **more than half of enterprise GenAI projects were abandoned after proof of concept by the end of 2025** — undone by poor data quality, weak risk controls, unclear value, and runaway cost. Notice that none of those failures are about the model; they are failures of the operating model around it. This guide lays out six layers — demand, models, grounding, guardrails, delivery, and LLMOps — over a governance spine that turns a promising pilot into a governed, Tier-1 capability. Read it as an assessment: see where each layer is strong, where it's being skipped, and what to close before you scale.

- **Layer 1 · Demand — start with the business, not the model**
- **Layer 2 · Models — treat models as a portfolio**
- **Layer 3 · Grounding — your data is the differentiator**
- **Layer 4 · Guardrails — safety is a layer, not a feature**
- **Layer 5 · Delivery — meet users inside their workflow**
- **Layer 6 · LLMOps — run it like any Tier-1 platform**

Tom Ward
Enterprise Architect · Cloud & AI

linkedin.com/in/tomward
July 2026

LAYER 1 Demand

"Start with the business, not the model"

Most enterprise LLM programs fail before a single token is generated — they fail at intake. Without a structured demand process, the portfolio fills with whatever the loudest sponsor wants, pilots multiply without owners, and the platform team drowns in one-off requests. Demand management is where an LLM program earns the right to be called an enterprise capability.

The mechanism is familiar to any architect: a lightweight intake funnel that scores every candidate use case on two axes — business value (revenue, cost, risk reduction, cycle time) and delivery risk (data sensitivity, error tolerance, regulatory exposure, integration complexity). High value / low risk goes first. Low value / high risk gets a polite no, in writing.

KEY DECISIONS

- Who owns intake — a federated AI council or a central CoE?
- What error tolerance does each use case actually have? Drafting an email tolerates mistakes; adjudicating a claim does not.
- Which use cases are assist (human reviews output) vs. automate (output acts directly)? Price the risk accordingly.

WATCH OUT FOR

- × "Pilot sprawl" — dozens of proofs-of-concept, zero production deployments, no kill criteria.
- × Scoring only value and ignoring feasibility — the flashiest use case is often the least grounded in available data.
- × Letting the tool choose the problem ("we bought licenses, find uses for them").

LAYER 2 Models

"Treat models as a portfolio"

The model layer changes faster than any technology an enterprise has ever managed — capability leaders leapfrog each other quarterly and prices drop by an order of magnitude within a year. Architecting around any single model is a guaranteed rewrite. The durable asset is not the model; it is the **catalog and the abstraction that lets you swap models without re-platforming**.

Buy vs. build vs. fine-tune is a per-use-case decision, not an enterprise standard. For most workloads, a frontier API model with good retrieval beats a fine-tuned smaller model — at a fraction of the engineering cost. Fine-tuning earns its keep for narrow, high-volume, latency-sensitive tasks with stable requirements.

KEY DECISIONS

- API consumption vs. private hosting — driven by data residency, latency, and cost-at-scale, not fashion.
- Maintain an approved-model catalog with tiers (frontier / workhorse / edge) mapped to use-case risk classes.
- Contract for model deprecation: what happens when a provider retires the version you validated?

WATCH OUT FOR

- × Hard-coding one vendor's SDK across every application — the lock-in is in the integration, not the model.
- × Fine-tuning to fix problems that are actually retrieval or prompting problems.
- × Benchmarking on public leaderboards instead of your own eval set.

LAYER 3 Grounding

"Your data is the differentiator"

Every competitor can call the same model APIs. What they cannot call is your loan history, your claims archive, your product telemetry. Retrieval-augmented generation (RAG) is how that proprietary context reaches the model — and it is where most of the real engineering in an enterprise LLM system lives: document pipelines, chunking strategy, embedding refresh, hybrid search, re-ranking.

The non-negotiable architectural rule: **retrieval must enforce the same entitlements as the source systems**. If a user cannot open the document in SharePoint, the RAG pipeline must not surface its contents in an answer. Access control at query time — not at indexing time alone — is what separates a compliant design from a data-breach headline.

KEY DECISIONS

- Which corpus is authoritative for each use case, and who owns its freshness SLA?
- Vector store selection: a dedicated engine vs. vector capability in the database you already run.
- Document-level security trimming at query time — mandatory for anything beyond public content.

WATCH OUT FOR

- × Indexing everything once and never refreshing — stale retrieval quietly poisons answer quality.
- × Chunking by page count instead of semantic structure; garbage chunks in, hallucinations out.
- × Copying entitled data into an unsecured index — the most common GenAI audit finding.

LAYER 4 Guardrails

"Safety is a layer, not a feature"

Guardrails are an architectural tier that sits between every model and every user — input filtering on the way in (prompt-injection shields, PII detection, jailbreak screening) and output filtering on the way out (content safety, grounding checks, policy enforcement). Treating them as per-application features guarantees inconsistency; treating them as a shared layer makes safety auditable and reusable.

The threat model is genuinely new. Prompt injection — untrusted content instructing the model to ignore its rules — has no perfect defense today, which is why layered controls plus least-privilege design (the model can only reach what its user can reach) matter more than any single filter. OWASP's Top 10 for LLM Applications is the fastest way to brief a security team that is new to the space.

KEY DECISIONS

- Centralized guardrail service vs. per-app libraries — centralize the policy, distribute the enforcement.
- Which policies are code (hard blocks) vs. review queues (human judgment)?
- Red-team cadence: who attacks your own assistants, and how often?

WATCH OUT FOR

- × Assuming the model provider's safety training is sufficient for your regulatory context.
- × Giving an agent write-access to systems its human user cannot touch.
- × Logging so little that you cannot reconstruct an incident after the fact.

LAYER 5 Delivery

"Meet users inside their workflow"

Adoption is a delivery-architecture problem. A standalone chat portal gets a burst of curiosity and then a 90% drop-off; the assistant that lives inside the CRM, the IDE, the claims platform — where the work already happens — is the one that survives. Design for the workflow first and the conversation second.

The keystone component is an **LLM gateway**: a single choke point through which every application reaches every model. It centralizes authentication, rate limiting, cost attribution by team, model routing, guardrail enforcement, and logging. It is also your vendor-independence insurance — swap the model behind the gateway and no application changes.

KEY DECISIONS

- Embed in existing tools (Copilot-pattern) vs. build a destination app — default to embed.
- Gateway build vs. buy: API management platforms are growing this capability fast.
- Chargeback model: who pays for tokens, and how is usage attributed?

WATCH OUT FOR

- × Shipping a chatbot when the workflow needed a button — not every LLM feature is a conversation.
- × Letting each team wire directly to model APIs; you lose cost control and auditability on day one.
- × Ignoring latency budgets — a 20-second answer is a broken feature in an operational workflow.

LAYER 6 LLMOps

"Run it like any Tier-1 platform"

LLM systems are non-deterministic, which means traditional testing is necessary but not sufficient. The discipline that replaces certainty is **evaluation**: a versioned suite of representative cases, scored automatically (with LLM-as-judge where human labels are too expensive), run before every prompt change, model upgrade, or retrieval tweak — exactly like a regression suite gates a code release.

Prompts are code. Version them, review them, and deploy them through a pipeline — not by editing a string in production. Pair that with runtime telemetry (token cost per feature, latency percentiles, refusal rates, grounding scores) and drift detection, because the model provider's silent updates and your users' shifting behavior will both erode quality without announcing themselves.

KEY DECISIONS

- What is your golden eval set, and who curates it as the business changes?
- Cost governance: budgets and alerts per use case, caching strategy, model right-sizing.
- Rollback story: when a model upgrade degrades quality, how fast can you pin the old version?

WATCH OUT FOR

- × "It worked in the demo" as the acceptance criterion — no eval set means no change safety.
- × Cost surprises from agentic loops — one runaway agent can spend a month's budget overnight.
- × Monitoring infrastructure health but not answer quality.

Governance: The Spine Across All Six Layers

Governance is not a committee that meets monthly — it is a set of controls wired into the platform itself: a named accountable owner for every use case, an audit trail of every prompt and response, human-in-the-loop escalation where confidence is low or stakes are high, and value tracking honest enough to retire what is not working. NIST's AI Risk Management Framework provides vocabulary that risk officers and regulators already recognize; mapping your controls to it turns governance reviews from debates into checklists.

A 10-Point Readiness Checklist

- [ ] **1. Every production use case has a named business owner and a written error-tolerance statement.**
- [ ] **2. An intake process scores value vs. risk — and has actually said no to something.**
- [ ] **3. Applications reach models only through a gateway; direct API wiring is prohibited.**
- [ ] **4. Retrieval enforces source-system entitlements at query time.**
- [ ] **5. Input and output guardrails are a shared service, mapped to OWASP's LLM Top 10.**
- [ ] **6. Prompts and model versions are under version control and deployed via pipeline.**
- [ ] **7. A golden eval suite gates every prompt, model, or retrieval change.**
- [ ] **8. Token cost is attributed per use case, with budgets and anomaly alerts.**
- [ ] **9. Every prompt/response is logged and reconstructable for incident review and audit.**
- [ ] **10. Value is measured quarterly against the original business case — and at least one use case has been retired.**

Going Deeper: References

Sources worth your time, roughly in the order a program team should read them.

NIST AI Risk Management Framework

The de-facto governance vocabulary for US enterprises — map your controls to it and regulator conversations get dramatically easier. Includes the Generative AI Profile.

nist.gov/itl/ai-risk-management-framework

OWASP Top 10 for LLM Applications

The fastest way to brief a security team on LLM-specific threats: prompt injection, insecure output handling, data poisoning, excessive agency.

owasp.org/www-project-top-10-for-large-language-model-applications

Microsoft Azure Well-Architected Framework — AI Workloads

Concrete design guidance for production AI workloads: reliability, cost, and operations patterns that translate beyond Azure.

learn.microsoft.com/en-us/azure/well-architected/ai/get-started

Google Secure AI Framework (SAIF)

A security practitioner's view of the AI stack — useful complement to OWASP for building your guardrail layer.

safety.google/cybersecurity-advancements/saif

MITRE ATLAS

Adversarial threat landscape for AI systems, in the familiar ATT&CK style — the red team's map for your Layer 4.

atlas.mitre.org

Anthropic — Building Effective Agents

Clear-eyed engineering guidance on when simple LLM workflows beat autonomous agents; directly informs Layer 5 and 6 design choices.

anthropic.com/research/building-effective-agents

AWS Well-Architected Framework — Generative AI Lens

AWS's counterpart to the Azure guidance: lifecycle, cost, and operations best practices for GenAI workloads.

docs.aws.amazon.com/wellarchitected/latest/generative-ai-lens

McKinsey — The State of AI

The adoption survey your executives have already read — useful for meeting the business case where it starts.

mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

Stanford HAI — AI Index Report

The annual evidence base: adoption, cost curves, and capability trends — invaluable for the business-case slide.

aiindex.stanford.edu

About the Author

Tom Ward is an Enterprise Architect with 20+ years across Fortune 500 financial services, retail, and government — including Wells Fargo and Bank of America — specializing in EA governance, reference architectures, hybrid cloud, and legacy modernization, with recent hands-on GenAI and mobile delivery work.

Let's connect: linkedin.com/in/tomward