INSTITUTIONAL AI · MAINTAINED REFERENCE

The AI Transformation Field Guide

From AI pilots to operational transformation.

Version v2026.09.1
Updated September 11, 2026
https://erikcaldwell.com/field-guide/

Erik Caldwell

About this document

The AI Transformation Field Guide is a maintained, ungated reference for leaders moving state and local government, public authorities, and the companies that serve them from AI experiments into governed, measurable production. It covers the middle of the work that pilots skip: operating models, agents and their controls, data and architecture, security and identity, governance, economics and measurement, and then a plan for putting the catalog to work.

It is written for executives and senior operators who have to decide what reaches production, what it may touch, and how they will know it worked. Every topic is written at executive length, ten to fifteen minutes each, and cites primary sources rather than vendor material.

This is the PDF edition of version v2026.09.1, updated September 11, 2026. The canonical, always-current text lives at https://erikcaldwell.com/field-guide/, where each section is also available as Markdown and where the whole guide can be searched. The latest edition of this PDF is always at https://erikcaldwell.com/field-guide/downloads/, and every change between versions is recorded at https://erikcaldwell.com/field-guide/changelog/.

Corrections are welcome through the contact form at https://erikcaldwell.com/contact/.

Views are my own.

Contents

Start Here

The Topic Catalog

START HERE

Executive Summary

Last updated September 11, 2026 · 3 min read · https://erikcaldwell.com/field-guide/executive-summary/

Purpose of This Document

This is a working curriculum, not a briefing memo. It exists to get one executive (leading enterprise AI transformation across a large technical and professional workforce) to a level of technical fluency sufficient to do five things without relying on a translator:

set AI investment and architecture strategy; challenge vendors and internal technical teams with informed, specific questions; redesign how software gets built and how knowledge work gets done; manage the security, governance, and compliance exposure that agentic AI introduces; and measure whether any of it is actually producing business value rather than activity.

It is explicitly not a path to becoming a machine learning engineer or researcher. Model architecture, training mechanics, and research-frontier mathematics are out of scope by design. What is in scope is everything a technically serious executive needs to make defensible decisions and ask the questions that make technical teams uncomfortable in a useful way.

How This Document Is Organized

The bulk of the document is a topic catalog: 80 topics across 20 domains, each written to a fixed template (definition, why it matters, what to understand, questions to ask your team, technologies to know, recommended learning, time investment), plus a 21st domain (an Emerging Technology Radar) covering 28 developments that are real but not yet decision-relevant, tracked so they don't arrive as a surprise.

Every topic carries a priority tier, and the tiers are meant to be taken literally:

Must Understand: you should be able to explain this to a peer executive and to your board without notes, and you should be uncomfortable delegating the underlying decision entirely.

Should Understand: you need working knowledge deep enough to evaluate a recommendation from your team, but the day-to-day mechanics are theirs to own.

Monitor: track directionally. Don't build expertise here yet; build the habit of checking back on it quarterly.

54 topics are Must Understand, 18 are Should Understand, and 8 are Monitor: reflecting a deliberately curated catalog rather than an exhaustive one. Everything included here already cleared a bar; if it's in this document at all, it's more likely to matter than not. A separate synthesis section, the 90-Day Executive AI Learning Plan, sequences this material into a realistic 3-5 hour/week study plan across three phases: Foundations (Days 1-30), Architecture & Transformation (Days 31-60), and Scaling the Enterprise (Days 61-90). If you read nothing else in this document, read that plan and the Top 25 Concepts cheat sheet that follows it.

Three Questions This Document Is Built to Answer

What do I personally need to understand? The topic catalog and the 90-Day Plan, in that order. Start with the Must Understand tier in Domains 1-4 (strategy, agents, Model Context Protocol (MCP), software engineering): that is where the largest, least reversible decisions sit in 2026.

What capabilities does my organization need to build? The Enterprise AI Capability Model at the end of this document maps 15 capability areas (from AI platform architecture to workforce transformation) across four maturity levels. Use it as a scoring instrument, not a reading list: most organizations at your scale are Level 1 or 2 in most areas, and that is normal, not alarming, in September 2026.

What should I watch now versus later? The Executive Technology Radar table converts the topic catalog and the emerging-technology radar into a single view: current importance, 2-3 year potential, and a recommended action: Adopt, Build Capability, Experiment, Understand, or Monitor. Not everything in this document earns "Adopt." Several genuinely promising developments (autonomous coding agents with merge authority, agent-to-agent commerce, fully autonomous business departments) are still speculative enough that the correct action is to watch, not to fund.

A Note on Certainty

This document distinguishes, deliberately and repeatedly, between what is established (DevOps Research and Assessment (DORA)'s research on AI-assisted engineering, National Institute of Standards and Technology (NIST)'s risk management framework, the Open Worldwide Application Security Project (OWASP) Large Language Model (LLM) Top 10), what is directionally clear but still moving (the EU AI Act's implementation timeline, agent identity standards, MCP governance), and what is genuinely speculative (autonomous departments, agent-to-agent commerce, embodied AI at enterprise scale). Where a claim is a prediction rather than a documented fact, it is flagged as such. Vendor marketing, consultant thought-leadership, and SEO content were deliberately excluded as sources in favor of primary documentation, standards bodies, peer-reviewed and preprint research, and named engineering organizations disclosing their own data. Every factual claim in this document links to its source.

Enterprise AI Strategy & Operating Model

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/enterprise-ai-strategy-and-operating-model/

1.1 AI Operating Models: Centralized vs Federated vs Hub-and-Spoke (incl. Center of Excellence)

Priority: Must Understand

Executive Definition: An AI operating model defines who builds AI capability, who decides what gets funded, and who is accountable when something breaks. Centralized means one team owns AI end-to-end; federated means each business unit builds its own with light shared standards; hub-and-spoke (the most common enterprise pattern) puts a central team in charge of platform, governance, and hard problems while embedded "spoke" practitioners sit inside product teams and ship. A Center of Excellence (CoE) is the hub in that pattern, not a synonym for the whole model.

Why It Matters: At meaningful organizational scale, the wrong model produces either duplicated tooling and inconsistent risk controls (pure federation) or a bottlenecked hub that business units route around with shadow AI (pure centralization). DORA's 2025 research found platform investment is a prerequisite for AI value capture (90% of high-performing organizations had adopted an internal platform before AI paid off) which is effectively an argument for a hub that owns platform and guardrails regardless of how build work is federated.

What I Need to Understand:

- The distinction between owning the platform/guardrails (hub) vs. owning delivery (spoke): most real deployments split these, they aren't mutually exclusive
- Where governance, model access, security review, and cost controls sit vs. where use-case selection and build work sit
- How the model changes as adoption matures (early: centralize to move fast and control risk; later: federate delivery once guardrails are proven)
- The failure modes of each: centralized = bottleneck and shadow AI; federated = duplicated spend, inconsistent risk posture, no shared learning; hub-and-spoke = unclear accountability if the split isn't explicit
- How this interacts with your existing platform engineering org: a new "AI CoE" that duplicates platform engineering's job is a red flag

Questions I Should Be Able to Ask My Team:

1. When two business units want conflicting model configurations or exceptions to our guardrails, who has the authority to say no, and has that ever actually happened?
2. What decisions does the central AI team make versus what decisions do product teams make on their own: can you show me the line, not just describe it?

3. If we shut down the central AI team tomorrow, what would silently break, and what would nobody notice was gone?

Technologies / Standards / Companies to Know: Team Topologies (platform team / stream-aligned team vocabulary), DORA AI Capabilities Model, internal developer platforms (Backstage, Port).

Recommended Learning:

- DORA AI Capabilities Model report: the seven organizational capabilities that determine whether AI adoption actually pays off; grounds the "hub owns the enabling conditions" argument.
- Team Topologies: the book's site: the underlying team-type vocabulary (stream-aligned, platform, enabling, complicated-subsystem) that most hub-and-spoke designs borrow from, whether they cite it or not.
- Enterprise AI Operating Model: Hub-and-Spoke, Federated, or Centralized?: a practical comparison; treat as a starting taxonomy, not a settled standard, since no standards body defines these terms consistently.

Time Investment: 2-3 hours

1.2 AI Portfolio Management, Use-Case Prioritization & Build vs Buy vs Configure

Priority: Must Understand

Executive Definition: Portfolio management is treating your AI initiatives as a funded, prioritized set of bets rather than a scattered list of pilots: with the same rigor you'd apply to a product or capital portfolio. Build vs. buy vs. configure is the decision, per use case, between building custom (highest cost/control), buying a vendor product (fastest, least differentiated), or configuring a platform/agent framework around your data (the now-common middle path).

Why It Matters: McKinsey's 2025 global survey found only 39% of organizations attribute any measurable EBIT impact to AI and just 6% qualify as "high performers" (5%+ EBIT impact): the gap is overwhelmingly a portfolio-discipline problem, not a technology problem: too many low-value pilots, unclear ownership, no kill criteria. The same survey found coding agents are shifting the calculus: 32% of respondents report forgoing a software purchase because an internal team could build it with AI assistance: inflating the "build" option's apparent attractiveness in ways that need scrutiny (build still carries maintenance, security, and model-drift costs a vendor absorbs).

What I Need to Understand:

- The difference between a pilot (proof of concept, no production commitment) and a scaled deployment, and what percentage of your pilots are actually converting
- A defensible scoring framework for use cases: value (revenue, cost, risk reduction) against feasibility (data readiness, integration complexity, change-management load): and that no single framework is industry-standard

- Why "buy" often wins even when "build" looks cheaper up front: ongoing model updates, security patching, and support are recurring costs a vendor amortizes across many customers
- What "configure" actually means for agentic systems: buying a platform (e.g., an agent framework or Copilot-style product) and building thin business logic on top, versus building the whole stack
- Kill criteria: what causes you to sunset a pilot, and whether anyone currently owns that decision

Questions I Should Be Able to Ask My Team:

1. Of our current AI initiatives, how many have an assigned business owner accountable for a measured outcome, versus how many are "IT is running a pilot"?
2. For this build-vs-buy decision, what's the fully loaded 3-year cost of building (including model updates, security review, and the engineer-hours to maintain it) not just the initial build estimate?
3. Which of our current pilots have we killed in the last two quarters, and why did we kill them?

Technologies / Standards / Companies to Know: McKinsey/BCG/Bain AI value-tracking research (treat as industry survey data, not standards), internal AI platforms (Backstage-style catalogs), vendor agent platforms (Microsoft Copilot Studio, Salesforce Agentforce, ServiceNow AI Agents) as reference points for "configure."

Recommended Learning:

- The State of AI: Global Survey 2025 (McKinsey): primary survey data on adoption-to-impact gap, scaling rates, and the build/buy shift from coding agents; read the actual survey, not summaries of it.
- DORA 2025: State of AI-assisted Software Development: grounds why "build" now looks more feasible for internal tools, and where that confidence is and isn't warranted.

Time Investment: 2-3 hours

1.3 Multi-Model Strategy & Vendor Dependency

Priority: Must Understand

Executive Definition: A multi-model strategy means your organization's AI systems can run on more than one foundation model provider (e.g., Anthropic, OpenAI, Google) without a full rebuild: achieved through abstraction layers, model routing, and portable interfaces like the Model Context Protocol (MCP). The alternative is single-vendor dependency: faster to start, but it concentrates pricing power, outage risk, and roadmap risk in one company's hands.

Why It Matters: Model quality, price, and capability leadership have changed hands repeatedly across Anthropic, OpenAI, and Google over the past two years, and that volatility is a documented pattern, not a one-off: locking your architecture to one model's API means re-architecting every time the leaderboard shifts or a vendor changes pricing or terms. At the same time, genuine multi-model portability has real engineering cost (prompt behavior, tool-calling conventions, and safety behavior differ across models), so "multi-model everywhere" is itself a cost decision, not a free hedge.

What I Need to Understand:

- Where in your architecture model choice is actually abstracted (a routing/gateway layer) versus where it's hard-coded into application logic: most enterprises overstate their real portability
- The Model Context Protocol (MCP), an open standard originated by Anthropic in late 2024 and since adopted by OpenAI, Google, Microsoft, and others for connecting models to tools and data: the closest thing to an interoperability standard this space has, though it is barely two years old and still evolving
- Supply-chain risk in the LLM sense: OWASP's LLM Top 10 flags "Supply Chain" (LLM03) as a top-tier risk category, covering third-party model, plugin, and data dependencies
- The real cost of multi-model: different models have different tool-calling behavior, safety refusal patterns, and prompt sensitivity: testing and maintaining behavior parity across models is nontrivial ongoing work, not a one-time integration
- When single-vendor is actually the right call (regulated data residency, a specific model's capability lead on your workload) versus when it's just inertia

Questions I Should Be Able to Ask My Team:

1. If our primary model provider raised prices 40% tomorrow or had a multi-day outage, what would happen to our production systems, and how long would switching actually take?
2. Where does model choice live in our codebase: is it a config value or is it baked into prompts and business logic across dozens of services?
3. Are we paying the ongoing cost of testing multiple models for behavioral parity, or did we buy "multi-model support" from a gateway vendor and never actually validate it?

Technologies / Standards / Companies to Know: Model Context Protocol (MCP), OWASP Top 10 for LLM Applications, model gateway/router products (e.g., LiteLLM, Portkey, cloud-native model gateways from AWS/Azure/GCP), Anthropic, OpenAI, Google DeepMind, Meta.

Recommended Learning:

- Introducing the Model Context Protocol (Anthropic): the original announcement and rationale for the standard now used across vendors.
- One Year of MCP (Model Context Protocol blog): an honest primary-source retrospective on adoption, maturity, and what's still unresolved in the spec.
- OWASP Top 10 for LLM Applications 2025: see LLM03 (Supply Chain) and LLM06 (Excessive Agency) specifically for vendor-dependency risk framing.

Time Investment: 2-3 hours

1.4 AI Maturity Models & Transformation Governance

Priority: Should Understand

Executive Definition: An AI maturity model is a staged framework (typically 4-5 levels, from ad hoc experimentation to fully governed, scaled deployment) used to assess how far an organization's AI capability has actually progressed, as distinct from how much it has spent or piloted. Transformation governance is the decision-rights and oversight structure (steering committees, risk gates, funding reviews) that moves an organization through those stages deliberately rather than by accident.

Why It Matters: McKinsey's 2025 survey data (37% attributing any EBIT impact to AI, only 6% "high performers") is itself evidence that most enterprises are stalled at pilot maturity regardless of spend: maturity models exist precisely to diagnose why, since the blockers at each stage (data readiness, then talent, then process redesign, then governance) are different problems requiring different interventions. Treat any single vendor's maturity model as a diagnostic lens, not a certification: there is no ISO or NIST-equivalent standard for "AI maturity" the way there is for, say, information security (the ISO/IEC 27001 standard, jointly published by the International Organization for Standardization and the International Electrotechnical Commission (IEC)).

What I Need to Understand:

- No maturity model is neutral: most are published by consultancies or vendors selling the services to move you up their own ladder; use them to structure a conversation, not as an external audit
- The common stage pattern across most models: (1) experimentation/pilots, (2) scaled pockets, (3) integrated into core processes, (4) governed and measured enterprise-wide, and that most organizations, including sophisticated ones, sit at stage 1-2
- Governance maturity is a distinct axis from technical maturity: an org can have advanced models in production with no risk review process, which is its own red flag (see Acceptable Use Policies topic)
- What "transformation governance" should concretely produce: a funding gate, a risk/security review gate, and a measurement standard applied consistently across initiatives: not a steering committee that meets and produces no decisions

Questions I Should Be Able to Ask My Team:

1. Which specific gate did our last three AI initiatives fail to clear before reaching production, and did any actually get stopped, or does everything eventually ship?
2. If you had to justify our maturity stage using only evidence (production usage numbers, measured outcomes, governance artifacts) not intentions, what would you point to?
3. Who besides IT/engineering sits on our AI governance body, and what's the last decision they actually made that changed a project's direction?

Technologies / Standards / Companies to Know: ISO/IEC 42001 (AI management systems: the closest thing to a certifiable standard in this space), NIST AI Risk Management Framework (RMF) (risk-focused, not maturity-staged, but the closest US government reference), McKinsey/Gartner maturity frameworks (useful, vendor-authored).

Recommended Learning:

- The State of AI: Global Survey 2025 (McKinsey): use as maturity evidence: adoption vs. impact gap by industry and company size.
- AI Risk Management Framework (NIST): the US government's govern/map/measure/manage structure; not a maturity ladder but the standard reference for what "governed" should mean at your top stage.

Time Investment: 1 hour

Agentic AI

Last updated September 11, 2026 · 15 min read · https://erikcaldwell.com/field-guide/agentic-ai/

2.1 What Is an AI Agent: Loops, Reasoning, Tool Use & Function Calling

Priority: Must Understand

Executive Definition: An AI agent is a system where a language model runs in a loop: observing state, reasoning about what to do next, calling external tools (functions, APIs, databases), and using the results to decide its next step: until it determines the task is done or hits a stopping condition. Function calling/tool use is the mechanical layer that lets the model request a specific action with structured arguments instead of just producing text. This is architecturally different from a single request-response LLM call.

Why It Matters: Nearly every "agent" product your teams build or buy is this loop with different guardrails around it. Understanding the loop lets you distinguish genuine agentic automation from a chatbot with a marketing label, and lets you ask precise questions about where the loop can go wrong (infinite loops, wrong tool selection, cost per turn).

What I Need to Understand:

- The ReAct pattern (reason → act → observe → repeat) is the conceptual ancestor of nearly all current agent loops (Yao et al., 2022)
- "Tool use" and "function calling" are the same underlying mechanism: the model outputs a structured call, your code executes it, the result is fed back into context
- An agent's reliability is bounded by three things: how well tools are described/scoped, how good the model is at picking the right one, and how the harness handles errors and retries
- More tools available to a model is not automatically better: overlapping or poorly documented tools degrade selection accuracy
- Cost and latency scale with loop iterations, not just input size: a runaway loop is a cost and safety incident, not just an inconvenience

Questions I Should Be Able to Ask My Team:

1. How many tools does this agent have access to, and how do we know the model reliably picks the right one as that number grows?
2. What is the maximum number of loop iterations or tool calls before we force a stop, and what happens when that ceiling is hit mid-task?
3. When a tool call fails or returns an unexpected result, does the agent retry blindly, escalate to a human, or silently proceed with bad data?

Technologies / Standards / Companies to Know: Anthropic Claude tool use/Agent SDK (Software Development Kit), OpenAI function calling/Assistants & Agents SDK, ReAct (academic origin), LangGraph, Model Context Protocol (as the interop layer for tools: see separate entry)

Recommended Learning:

- ReAct: Synergizing Reasoning and Acting in Language Models: the original academic framing of the reason-act-observe loop; still the reference point for how agent loops are described.
- Tool use with Claude: Claude Platform Docs: primary-source mechanics of how tool calls are defined, executed, and fed back into an agent loop.
- Building Effective AI Agents: Anthropic: draws the line between an "augmented LLM" (one call plus tools) and a true autonomous agent loop.

Time Investment: 1 hour

2.2 Agent Architectures: Single-Agent, Multi-Agent & Orchestration Patterns

Priority: Must Understand

Executive Definition: Most production agent systems are not one model freelancing: they're composed from a small set of named patterns: prompt chaining, routing, parallelization, orchestrator-worker (a lead agent delegates to subagents and synthesizes results), and evaluator-optimizer (one model critiques another's output). A single autonomous agent, operating open-endedly, is the least common and highest-risk pattern in practice. Multi-agent systems trade higher token cost and coordination complexity for better task decomposition on broad, parallelizable work.

Why It Matters: Vendors and internal teams will describe almost anything as "multi-agent" because it sounds more advanced; knowing the actual patterns lets you evaluate whether the added complexity (and cost: Anthropic reports multi-agent research systems can use roughly 4x the tokens of a single chat interaction) is buying real capability or just overhead. This is the single highest-leverage vocabulary for challenging architecture decisions in this domain.

What I Need to Understand:

- Workflows (predefined code paths orchestrating LLM calls) and agents (the LLM decides its own path) are a spectrum, not a binary: most reliable production systems today lean toward workflows with agentic pieces, not fully open-ended agents
- Orchestrator-worker (a "lead" agent spawning subagents with separate context windows) is the dominant multi-agent pattern for research/analysis-style tasks; it works well for breadth-first, parallelizable problems and poorly for tasks requiring shared, tightly coupled state
- Multi-agent systems are harder to debug and more expensive to run than single-agent systems, and Anthropic's own writeup is explicit that they used them only after establishing the added performance justified the cost

- "Evaluator-optimizer" (model critiques another model's draft) is a distinct, useful pattern for quality-sensitive output, separate from task decomposition
- There is no universal "best" architecture: the right pattern is a function of whether subtasks can be parallelized and whether they need shared context

Questions I Should Be Able to Ask My Team:

1. Which named pattern is this system actually using, and why was a simpler single-agent workflow ruled out?
2. What is the token/cost multiplier of this multi-agent design versus a single-agent equivalent, and did we measure whether the accuracy gain justifies it?
3. How do subagents share state or hand off results, and what happens when two subagents produce conflicting outputs?

Technologies / Standards / Companies to Know: Anthropic (orchestrator-worker research system), OpenAI Agents SDK, LangGraph, Microsoft AutoGen/Semantic Kernel, CrewAI

Recommended Learning:

- Building Effective AI Agents: Anthropic: the primary-source taxonomy of workflow/agent patterns (chaining, routing, parallelization, orchestrator-workers, evaluator-optimizer).
- How we built our multi-agent research system: Anthropic: concrete engineering tradeoffs, token cost multipliers, and when Anthropic's own team chose (and avoided) multi-agent design.

Time Investment: 1 hour

2.3 Long-Running, Computer-Use & Browser Agents

Priority: Should Understand

Executive Definition: These are agents that act directly on a computer's GUI or browser (clicking, typing, navigating) rather than calling clean APIs, and that run for extended periods (minutes to hours) rather than a single exchange. Anthropic's "computer use" and OpenAI's "Operator"/computer-using-agent (CUA) are the two production offerings; both work by having the model view screenshots and issue mouse/keyboard actions in a loop.

Why It Matters: Computer-use agents are the path to automating legacy systems and workflows with no API (which is most of the "professional/knowledge worker" surface area at a mid-size enterprise) but current benchmark data shows this capability is meaningfully behind API-based tool use in reliability, and leadership should not greenlight unattended production deployment on this basis alone.

What I Need to Understand:

- On OSWorld 2.0, a long-horizon real-computer-use benchmark, the best frontier models complete only ~13–21% of tasks correctly on strict scoring as of mid-2026, and completion rates collapse toward zero as task length increases (arXiv 2606.29537): this is lab/benchmark evidence, not

marketing

- Failure modes are specific and dangerous for unattended use: agents lose track of task constraints over long horizons, spend little effort on self-correction, and the benchmark documented real incidents of credential leakage and unauthorized system changes
- "Long-running" also refers to agentic coding/analysis sessions (not just GUI control) that persist for hours: these need the same session-management, checkpointing, and interruption-handling discipline as GUI agents
- Browser agents (distinct from full computer-use) operate inside a sandboxed browser context, which narrows the attack/failure surface relative to full desktop control
- Human oversight requirements should scale with task irreversibility and duration, not be uniform across all computer-use tasks

Questions I Should Be Able to Ask My Team:

1. What benchmark or internal evaluation did we run before allowing this computer-use agent to touch a production system, and what was its measured success rate on tasks representative of our real workflows?
2. What is the blast radius if this agent takes a wrong action mid-task (can it modify financial records, send external communications, or delete data) and is that reversible?
3. For long-running sessions, how do we checkpoint progress so a failure at hour three doesn't require starting over, and who is notified when the agent stalls or goes off-task?

Technologies / Standards / Companies to Know: Anthropic Computer Use (Claude), OpenAI Operator / Computer-Using Agent (CUA), OSWorld benchmark, Claude Agent SDK

Recommended Learning:

- Computer use tool: Claude Platform Docs: primary-source mechanics and current limitations as documented by Anthropic.
- Computer-Using Agent: OpenAI: OpenAI's own framing of the CUA model behind Operator, including stated limitations.
- OSWorld 2.0 (arXiv 2606.29537): the most current independent benchmark data on long-horizon computer-use reliability; explicitly concludes agents are "far from solving long-horizon professional computer use."

Time Investment: 2-3 hours

2.4 Agent Memory & Persistent State

Priority: Should Understand

Executive Definition: LLMs have no memory beyond the tokens in their current context window; "agent memory" is the engineering layer built on top: compaction (summarizing history and discarding detail), external note-taking (the agent writes progress to a file or database it re-reads), retrieval systems that

fetch relevant past information on demand, and dedicated memory stores that persist facts across sessions. There is no single standard implementation; this is an active area of custom engineering, not a solved commodity feature.

Why It Matters: Context window limits are the practical ceiling on how long or complex a single agent task can run without deliberate memory engineering, and getting this wrong is a leading cause of agents that "forget" earlier constraints, contradict earlier decisions, or silently drop requirements on long tasks: a direct quality and trust risk for anything customer-facing or high-stakes.

What I Need to Understand:

- Context window is working memory, not storage: anything not explicitly persisted (to a file, database, or summary) is gone once the window fills or the session ends
- Anthropic identifies three complementary techniques for long-horizon state: compaction (summarize and restart with compressed history), structured note-taking (external files like a running progress log the agent re-reads), and subagent architectures (offload exploration to agents with clean contexts, return only distilled summaries)
- "Just-in-time" context retrieval (fetching data by reference (file path, query) at the moment it's needed rather than front-loading everything) is now considered better practice than stuffing the context window upfront
- Persistent cross-session memory (the agent remembers a user or account across separate conversations) is a distinct, less mature capability from within-session state management, and raises its own data governance and consent questions
- Vendor "memory" features vary widely in what they actually persist, for how long, and with what visibility/deletion controls: treat vendor claims here skeptically and ask for specifics

Questions I Should Be Able to Ask My Team:

1. When this agent's task runs long, what specifically happens to earlier context: is it summarized, dropped, or written to external storage, and what's the risk of losing an early constraint?
2. What cross-session memory does this system retain about a user or account, where is it stored, who can see it, and how is it deleted on request?
3. Have we tested this agent on a task long enough to force compaction or context-window pressure, and did it still honor requirements stated early in the task?

Technologies / Standards / Companies to Know: Anthropic context engineering guidance, Amazon Bedrock AgentCore Memory, Mem0, LangGraph persistence/checkpointing

Recommended Learning:

- Effective context engineering for AI agents: Anthropic: primary-source explanation of compaction, note-taking, and sub-agent patterns for managing state across long tasks.

Time Investment: 1 hour

2.5 Agent Sandboxes, Observability, Rollback & Idempotency

Priority: Must Understand

Executive Definition: Because agents take autonomous, sometimes-irreversible actions, production deployments require the same operational discipline as any high-risk automated system: sandboxing (isolating the agent's execution environment from production systems), observability (logging every reasoning step and tool call so failures can be diagnosed after the fact, not just the final output), rollback (the ability to undo an agent's actions), and idempotency (designing actions so repeating them (due to retries) doesn't cause duplicate side effects, like double-charging a customer).

Why It Matters: This is the operational infrastructure that separates a demo from something you can safely run against real systems; without it, an agent error is invisible until a customer or auditor finds it, and a retried action can silently compound the damage. This is exactly the kind of engineering rigor an executive should expect teams to show evidence of before approving production autonomy.

What I Need to Understand:

- "Sandboxing" for agents (running in an isolated container/VM with scoped permissions) is a direct analog to least-privilege access control and should be treated as non-negotiable for any agent that can execute code or call write-access tools
- Observability for agents means tracing the full reasoning chain and every tool call, not just input/output: because failures are often in the middle of a multi-step process, and standard application logs don't capture "why" the agent chose an action
- Idempotency matters specifically because agents retry: a network timeout that causes a tool call to be re-issued must not double-book, double-charge, or double-send: this is a design requirement on the tools the agent calls, not on the agent itself
- Rollback/undo capability should be evaluated per-tool: some actions (sending an email, external API calls) are inherently non-reversible and need a different control (pre-approval) rather than a rollback plan
- This tooling category (agent-specific observability/tracing platforms) is still maturing and fragmented across vendors: there is no dominant standard yet, so expect to evaluate and possibly switch

Questions I Should Be Able to Ask My Team:

1. Can we reconstruct, after the fact, the full chain of reasoning and tool calls that led to a specific agent action: not just its final output?
2. Which of this agent's actions are idempotent, which are irreversible, and what's the control (retry-safe design vs. mandatory pre-approval) for each category?
3. What sandbox boundary is this agent running inside, and what's the actual blast radius if it's compromised or goes wrong: what could it touch that it shouldn't?

Technologies / Standards / Companies to Know: OpenTelemetry (GenAI semantic conventions), Anthropic/OpenAI agent tracing tooling, LangSmith, container/VM sandboxing (Docker, Firecracker-style microVMs), Amazon Bedrock AgentCore

Recommended Learning:

- Building Effective AI Agents: Anthropic: explicitly calls out sandboxed testing and guardrails as prerequisites for autonomous agent deployment.
- Computer use tool: Claude Platform Docs: documents Anthropic's own guidance on running computer-use agents in isolated, controlled environments.

Time Investment: 1 hour

2.6 Human Approval, Delegated Authority & Autonomy Levels

Priority: Must Understand

Executive Definition: This is the design question of how much an agent is allowed to do without a human checking first: ranging from full pre-approval of every action, to approval only for high-risk/irreversible actions, to full autonomy with post-hoc monitoring. Anthropic's own measurement research found this is not well captured by rigid tiers; what matters operationally is whether a human is in a genuine position to monitor and intervene, not whether a specific interaction pattern (e.g., "approve every step") is mandated.

Why It Matters: Getting this wrong in either direction is costly: over-gating creates enough friction that agents deliver no efficiency gain (a documented driver of the well-known finding that most enterprise GenAI pilots fail to show ROI), while under-gating creates real financial, legal, and safety exposure. This is a governance decision, not just an engineering one, and it belongs on your desk.

What I Need to Understand:

- Anthropic's internal telemetry (Claude Code usage, Oct 2025–Jan 2026) found experienced users move toward higher auto-approval rates over time (roughly 20% to 40%+) while simultaneously increasing how often they interrupt the agent: a shift from per-action approval to active monitoring, not a simple "trust more, watch less" pattern
- Mandating human approval on every single action does not reliably produce better safety outcomes and can just add friction without benefit: the effective lever is quality of visibility and ease of intervention, not the presence of a checkpoint
- Delegated authority should be scoped per action-class by reversibility and stakes (e.g., "can read customer records" vs. "can issue a refund"), not as a single global autonomy dial for the whole agent
- Well-designed agents increasingly ask clarifying questions themselves when uncertain, rather than guessing: this is a design signal worth requiring from vendors and internal teams, not just a nice-to-have

- "Autonomy level" frameworks differ by vendor and are not standardized industry-wide: treat any vendor's "Level 3 autonomous" claim as marketing shorthand until you see the actual approval logic

Questions I Should Be Able to Ask My Team:

1. For this agent, which specific action categories require human pre-approval, which allow autonomous execution with post-hoc audit, and what was the reasoning for that split?
2. Do we have real visibility into what the agent is doing while it runs (not just its final report), and could a person actually intervene mid-task if something looked wrong?
3. What happens when the agent itself is uncertain: does it ask a clarifying question, guess, or stop, and how often is each of those observed in practice?

Technologies / Standards / Companies to Know: Anthropic autonomy/oversight research, OpenAI Agents SDK (guardrails/handoffs), enterprise agent-governance platforms (Strata, etc.)

Recommended Learning:

- Measuring AI agent autonomy in practice: Anthropic: primary-source data and framework on approval rates, interruption patterns, and what effective human oversight actually looks like in deployed agents.

Time Investment: 1 hour

2.7 Where Agents Beat vs Lose to Deterministic Software

Priority: Must Understand

Executive Definition: Deterministic software (fixed code paths, explicit business logic) is faster, cheaper, fully predictable, and easier to audit, but only for tasks you can fully specify in advance. Agents earn their cost premium and unpredictability only on tasks with open-ended judgment, ambiguous inputs, or paths that can't be fully enumerated at design time. The core management error is applying agents to well-specified, high-volume, deterministic tasks (worse cost and reliability than code) or applying rigid deterministic pipelines to genuinely ambiguous tasks (constant manual exception-handling).

Why It Matters: This single judgment call: agent vs. workflow vs. plain code: is where most wasted enterprise AI spend and most avoidable production incidents originate; an executive who can push back on "let's make it an agent" by default is directly protecting both budget and reliability.

What I Need to Understand:

- Anthropic's own guidance is explicit: "find the simplest solution possible, and only increase complexity when needed": a fixed workflow is preferred whenever the task's steps and rules can be predetermined

- Deterministic code wins on: cost per execution at volume, latency, testability (you can write unit tests with known correct answers), and auditability (the logic is inspectable, not inferred from model weights)
- Agents win on: tasks with combinatorially many valid paths, unstructured input requiring judgment (interpreting a customer's freeform complaint), or environments that change too often to hardcode rules for
- The "hybrid" pattern (deterministic code handling the guaranteed steps, with an agent invoked only for the genuinely ambiguous decision point) is usually the right default, not a compromise
- The MIT-linked 2025 finding that roughly 95% of enterprise generative AI pilots failed to show measurable P&L return is frequently attributed to organizations deploying general-purpose agentic tools on tasks that needed workflow rigor, integration, and process redesign, not model capability

Questions I Should Be Able to Ask My Team:

1. Could this task be done with deterministic code plus a small number of well-defined exception cases, and if not, what specifically makes it too ambiguous to hardcode?
2. What would this cost and how would its error rate compare if we built it as a fixed pipeline instead of an agent, and did we actually run that comparison?
3. Where in this system is an agent doing something a rules engine or a straightforward API integration could do more cheaply and more predictably?

Technologies / Standards / Companies to Know: N/A (conceptual/architectural judgment, not a specific product category)

Recommended Learning:

- Building Effective AI Agents: Anthropic: states directly that the simplest workflow should be preferred and agentic complexity added only when justified.
- MIT report: 95% of generative AI pilots at companies are failing: widely-cited 2025 MIT-linked field data on enterprise GenAI pilot ROI failure, useful context for why the workflow-vs-agent decision matters financially. *(Treat the 95% figure as directionally important, not a precise statistic: press coverage varies in how it characterizes MIT's underlying methodology.)*

Time Investment: 30 minutes

MCP & Agent Interoperability

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/mcp-and-agent-interoperability/

3.1 Model Context Protocol (MCP): Architecture & Enterprise Adoption

Priority: Must Understand

Executive Definition: MCP is an open protocol, originally released by Anthropic in late 2024, that standardizes how AI applications connect to external data sources and tools: described by its own documentation as "a USB-C port for AI applications." It defines a host/client/server architecture: an AI application (host) runs clients that each connect to one MCP server, which exposes tools, data ("resources"), and reusable prompts in a standard format any compliant client can consume.

Why It Matters: MCP is rapidly becoming the default way enterprises connect internal systems (databases, ticketing, Customer Relationship Management (CRM), internal APIs) to AI agents instead of building bespoke integrations per vendor per tool: reducing integration cost, but also creating a new internal surface (every MCP server you stand up) that needs the same access control and lifecycle discipline as any internal API.

What I Need to Understand:

- MCP's architecture is host → client → server: your AI application is the host, it maintains one client connection per external system, and each MCP server exposes that system's tools/data in a standard schema: this is what lets "build once, integrate everywhere" work across Claude, ChatGPT, VS Code, and other MCP-compliant clients
- As of December 2025, Anthropic donated MCP's governance to the newly formed Agentic AI Foundation (AAIF), a Linux Foundation directed fund co-founded by Anthropic, Block, and OpenAI, with Google, Microsoft, AWS, Cloudflare, and Bloomberg also participating: this is a genuine multi-vendor governance shift, not a single-vendor project anymore
- MCP is still evolving quickly: the protocol's own 2026 roadmap lists transport scalability, an improved task/retry model, formal contributor governance, and enterprise auth (SSO-integrated) as still-in-progress priorities, not finished features: treat "enterprise-ready MCP" claims from vendors with that in mind
- For your organization, MCP adoption practically means: which internal systems get an MCP server, who can build/approve one, and how you prevent an uncontrolled sprawl of servers each with their own credentials and access scope (see the MCP Security entry)
- Despite fast momentum (Linux Foundation cites MCP among projects with extremely high install/download counts), MCP is roughly two years old as of this writing: it has real vendor commitment but not the multi-decade track record of protocols like HTTP or OAuth, and its authorization and enterprise-governance layers are explicitly still being built

Questions I Should Be Able to Ask My Team:

1. Which internal systems currently have an MCP server exposed, who approved each one, and is there a central inventory, or did this happen organically without a registry?
2. What authentication/authorization does each MCP server enforce, and is it consistent, or does each server implementer choose its own?
3. If MCP's governance or spec shifts meaningfully under the new Linux Foundation structure, what's our exposure: how many internal systems and vendor integrations depend on the current protocol version?

Technologies / Standards / Companies to Know: Model Context Protocol (modelcontextprotocol.io), Agentic AI Foundation / Linux Foundation, Anthropic, OpenAI, Block (goose), Microsoft, AWS Bedrock AgentCore, Google

Recommended Learning:

- Introduction: Model Context Protocol: the official spec's own framing of what MCP is and its host/client/server model.
- Donating the Model Context Protocol and establishing the Agentic AI Foundation: Anthropic: primary-source announcement of the December 2025 governance transfer and founding members.
- The 2026 MCP Roadmap: Model Context Protocol Blog: the maintainers' own statement of what's still unfinished (enterprise auth, governance structure, transport scaling).

Time Investment: 2-3 hours

3.2 MCP Security & Governance (internal marketplaces/catalogs, authorization, malicious servers)

Priority: Must Understand

Executive Definition: Because MCP servers can be published by anyone and consumed by an AI agent that then acts on the results, MCP introduces a distinct attack surface: a malicious or compromised server can return tool results containing hidden instructions that the underlying model treats as trusted context ("tool poisoning"), potentially causing the agent to leak data or invoke other tools it shouldn't. Enterprise governance in response centers on internal MCP registries/catalogs (a vetted, approved list of servers: analogous to an app store), strict authorization scoping, and treating every tool response as untrusted input, not just every user input.

Why It Matters: This is the same class of risk as supply-chain attacks in traditional software (a compromised dependency), except the "code" here is natural-language instructions embedded in data the model reads: a category that's genuinely harder to filter than a code scanner catches, and one your security team may not yet have tooling for. Getting ahead of this with an internal approval process is materially cheaper than remediating after an incident.

What I Need to Understand:

- "Tool poisoning" works because there's a trust gap between connect-time (when a human reviews a tool's stated description) and runtime (when the tool actually executes and returns data): a server can describe itself innocently and only inject malicious instructions in its live responses, per OWASP's documentation of the pattern
- Mitigations cluster into: requiring structured/fixed-schema tool outputs instead of free text, isolating high-privilege tools from any server sourced externally, enforcing access control at the tool/server layer (not relying on the model's system prompt as a security boundary), and allowlisting only pre-approved servers
- Enterprises are converging on internal MCP registries/marketplaces (a curated, security-reviewed catalog of approved servers) as the practical governance unit: the same pattern as internal app stores or approved-vendor lists, now applied to agent tools
- The MCP authorization specification (OAuth 2.1-based) defines how clients and servers should authenticate, but implementation is inconsistent across the ecosystem today: verifying a given server actually implements it correctly is a real due-diligence step, not a checkbox
- This is one of the fastest-moving and least standardized parts of the whole domain: expect the tooling (scanners, registries, policy engines) here to change significantly within 12-18 months, and design your governance process to be tool-agnostic

Questions I Should Be Able to Ask My Team:

1. Do we have an approved internal registry of MCP servers, or can any team or individual connect an agent to an arbitrary external MCP server today?
2. How do we defend against tool poisoning specifically: do we treat tool call responses as untrusted content requiring the same scrutiny as user input, or only screen the tool's stated description at connection time?
3. For any MCP server handling sensitive data, does it correctly implement the MCP authorization spec, and who verified that rather than took the vendor's word for it?

Technologies / Standards / Companies to Know: OWASP (MCP Tool Poisoning documentation), MCP Authorization spec, Invariant Labs, Kong/JFrog/other MCP registry vendors, Amazon Bedrock AgentCore Gateway

Recommended Learning:

- MCP Tool Poisoning: OWASP Foundation: the clearest primary-source explanation of the attack mechanism and standard mitigations.
- MCP Security Notification: Tool Poisoning Attacks: Invariant Labs: the original technical disclosure that brought this attack class to wide attention.
- Diving Into the MCP Authorization Specification: Descope: accessible breakdown of the OAuth 2.1-based authorization model and its current gaps.

Time Investment: 2-3 hours

3.3 Agent2Agent (A2A) Protocol & the Interoperability Landscape

Priority: Monitor

Executive Definition: A2A is an open protocol, originally developed by Google and now governed by the Linux Foundation, for letting independent AI agents (potentially built by different vendors, on different frameworks) discover each other's capabilities and communicate to delegate and coordinate tasks. It is explicitly complementary to MCP rather than competing with it: MCP standardizes how one agent connects to tools and data; A2A standardizes how separate agents talk to each other, using structured "Agent Cards" for capability discovery and cryptographic identity verification.

Why It Matters: This matters if your roadmap includes agents built by different vendors or business units needing to hand off work to each other (e.g., your internal procurement agent negotiating with a supplier's agent); for most mid-size enterprises this is still nearer to the adoption curve's early stages than MCP, and should be tracked rather than committed to as core infrastructure today.

What I Need to Understand:

- A2A reached a 1.0 stable specification and, per Linux Foundation reporting, surpassed 150 supporting organizations and shipped SDKs in five languages, with integration into Azure AI Foundry, Amazon Bedrock AgentCore, and Google's platforms within roughly a year of its Linux Foundation donation
- The stated division of labor is: MCP = agent-to-tool/data, A2A = agent-to-agent: in practice, a single system may use both simultaneously, MCP internally within one agent and A2A to talk to external or partner agents
- A2A is younger and has a narrower proven production footprint than MCP; adoption numbers reflect organizational support/SDK availability, which is a different (and weaker) signal than deep production usage at scale: treat vendor and press framing of "adoption" with that distinction in mind
- There is genuine, open disagreement in the field about whether a distinct inter-agent protocol is durably necessary long-term, versus agent-to-agent communication converging back onto MCP or HTTP/API-based patterns as the ecosystem matures: this is contested and unresolved, not settled
- Related but distinct efforts exist in this same "agent interoperability" space (e.g., Agent Payments Protocol for agent-initiated transactions): the standards landscape here is still actively consolidating, not finished

Questions I Should Be Able to Ask My Team:

1. Do we have an actual near-term use case requiring agent-to-agent communication across organizational or vendor boundaries, or is A2A adoption being proposed speculatively?
2. If we adopt A2A now, what's our exposure if the standard doesn't consolidate the way current momentum suggests: how much of our integration work would need to be redone?
3. Where does our use of MCP end and A2A begin in this design, and is that boundary actually necessary, or could the same coordination be done with MCP alone?

Technologies / Standards / Companies to Know: Agent2Agent (A2A) Protocol, Linux Foundation / Agentic AI Foundation, Google, Microsoft Azure AI Foundry, Amazon Bedrock AgentCore, Agent Payments Protocol (AP2)

Recommended Learning:

- A2A and MCP: A2A Protocol: the protocol's own primary-source explanation of how A2A and MCP are meant to divide responsibilities.
- A2A Protocol Surpasses 150 Organizations…: Linux Foundation: primary-source adoption figures and governance status, one year post-donation.

Time Investment: 1 hour

AI-Native Software Engineering

Last updated September 11, 2026 · 21 min read · https://erikcaldwell.com/field-guide/ai-native-software-engineering/

4.1 The Software Engineering Continuum: Autocomplete → Coding Assistant → Coding Agent → Agent-Directed Engineering → Autonomous Software Factory

Priority: Must Understand

Executive Definition: AI's role in software engineering has moved through distinct stages, each requiring a different operating model, not just a better tool. Autocomplete (predicts the next few lines) gave way to coding assistants (chat-based help inside the IDE), then to coding agents (given a task, they read the repo, write code, run tests, and iterate with limited supervision), then to agent-directed engineering (a developer directs multiple agents against specifications rather than writing code line-by-line), and finally toward the autonomous "software factory" (issue-to-PR pipelines with minimal human touch). Most enterprises today sit between stage 2 and stage 3; stages 4-5 are early and unevenly proven.

Why It Matters: Each stage changes what "developer productivity" means, where risk enters the system, and what skills matter: conflating stages leads to buying the wrong tools, setting the wrong KPIs, or assuming maturity the organization doesn't have. DORA's 2025 research frames AI as an "amplifier" of existing organizational practice, not a stage-skipping shortcut: teams with weak fundamentals get faster and worse simultaneously (dora.dev, 2025). Vendor marketing routinely describes stage-4/5 capability while shipping stage-2/3 tools, so leaders need the vocabulary to tell the difference.

What I Need to Understand:

- The five stages are not merely "more automation": each one shifts *who* verifies correctness, *when* review happens, and *what unit of work* the human hands off (a line, a function, a PR, a ticket, a backlog).
- Autonomy is task- and codebase-dependent, not organization-wide: a team can be at "coding agent" maturity on a well-tested microservice and stuck at "assistant" on a legacy monolith with no test coverage: asking "where are we on the continuum" without qualifying by system is meaningless.
- The "autonomous software factory" (issue in, PR out, no human in the loop) is still largely aspirational at enterprise scale in 2026; treat vendor claims of full autonomy as marketing until you see evidence in a comparable codebase (see Coding Agents topic).
- Adoption of the tools (assistants, agents) is near-universal: DORA reports ~90% of respondents use AI at work (dora.dev, 2025) and GitHub reports roughly 80% of new developers use Copilot within their first week (GitHub Octoverse 2025): but tool adoption is a poor proxy for which *stage* of the continuum a team actually operates at.

- Moving up the continuum requires investment unrelated to the AI tool itself: test automation, modular architecture, clear specifications, and CI/CD maturity: DORA explicitly ties AI value realization to platform engineering and loosely coupled architecture (dora.dev, 2025).

Questions I Should Be Able to Ask My Team:

1. For our top three product lines, which stage of the continuum are we actually operating at, and what's the evidence (not the tool license)?
2. What's blocking us from moving from "coding agent" to "agent-directed engineering" on this system: is it test coverage, architecture, specification quality, or trust?
3. When a vendor claims "autonomous" or "agentic" capability, what specifically is still human-reviewed, and what's the failure mode when the agent is wrong?

Technologies / Standards / Companies to Know: GitHub Copilot (assistant → agent), Anthropic Claude Code, OpenAI Codex, Cursor, Google Jules, Devin (Cognition): positioned as more autonomous end of spectrum; Factory.ai and similar "software factory" platforms represent the more speculative end.

Recommended Learning:

- DORA: State of AI-assisted Software Development 2025: the primary-source framing of AI as amplifier, not autopilot.
- DORA: Balancing AI tensions: moving from adoption to effective Software Development Lifecycle (SDLC) use: names the velocity, expertise, and workflow tensions across the continuum.
- GitHub Octoverse 2025: adoption and agent-generated PR data at platform scale.
- Factory.ai: From coding agents to software factories: a vendor's own articulation of the later stages (read critically, as a claims document, not a case study).

Time Investment: 1 hour

4.2 Coding Agents & Autonomous Coding Agents

Priority: Must Understand

Executive Definition: A coding agent is given a task description rather than a single prompt, and it independently reads the codebase, plans, writes code, executes commands (build, test, lint), observes the results, and iterates: closing its own feedback loop instead of returning one suggestion for a human to accept or reject. This is qualitatively different from autocomplete or chat assistance (earlier stages of the continuum above). "Autonomous" is a spectrum of how much of that loop runs without a human checkpoint, from agents that pause for approval at each tool call to agents that run a whole task end-to-end and only surface a finished pull request.

Why It Matters: Agents generate large volumes of working code fast, but they also generate large volumes of *wrong* code fast, and the org's ability to safely absorb that output (not the agent's raw capability) is usually the binding constraint. GitHub reports over 1 million pull requests created by its Copilot coding agent between May and September 2025 (GitHub Octoverse 2025), showing this is now

production-scale, not experimental. At the same time, a randomized controlled trial by Model Evaluation & Threat Research (METR) found experienced open-source developers took 19% *longer* on real tasks when using current AI tools, despite believing afterward they had been roughly 20% faster: a direct warning against trusting self-reported productivity data (METR, July 2025).

What I Need to Understand:

- Agent effectiveness depends heavily on task shape: well-specified, well-tested, well-scoped tasks in familiar codebases succeed far more often than ambiguous, cross-cutting, or poorly tested ones: GitHub's data shows agents adopted fastest in established, well-instrumented repositories (GitHub Octoverse 2025).
- The agent's "autonomy level" is configurable: how many tool calls it can make unsupervised, whether it can run destructive commands, whether it can push to protected branches, and that configuration is a governance decision, not just an engineering one.
- Perceived productivity and measured productivity diverge; the METR finding that developers *felt* faster while being *measurably slower* means self-reported velocity metrics from teams adopting agents should be treated skeptically absent objective data (METR, 2025).
- Agent output volume strains downstream processes (code review, CI, incident response) that were sized for human-paced change: see AI Code Review and CI/CD topics below.
- "Autonomous coding agent" claims should be evaluated against a specific, checkable task class, not treated as a general capability rating.

Questions I Should Be Able to Ask My Team:

1. On the tasks where we've deployed coding agents, what's our actual completion rate without human rework, and how do we measure that (not self-report)?
2. What can an agent do unsupervised in our environment today (can it merge, deploy, or touch production credentials) and who approved that boundary?
3. Have we independently measured cycle time and defect rate before/after agent adoption, or are we relying on developer sentiment surveys?

Technologies / Standards / Companies to Know: GitHub Copilot coding agent, Claude Code (Anthropic), OpenAI Codex, Cursor Agent, Google Jules, Devin (Cognition), OpenHands (open source).

Recommended Learning:

- METR: Measuring the impact of AI on experienced open-source developer productivity (see coverage: The Register, "AI coding tools make developers slower, study finds"): the RCT behind the 19%-slower / 20%-perceived-faster finding.
- Anthropic: Effective harnesses for long-running agents: engineering perspective on what makes agents reliable over long tasks.
- GitHub Octoverse 2025: scale data on agent-generated PRs in production.

Time Investment: 1 hour

4.3 Specification-Driven Development & Context Engineering for Code

Priority: Must Understand

Executive Definition: As agents take on more of the implementation, the developer's job shifts toward writing precise specifications and curating the context an agent needs to act correctly: repository-level instruction files, architecture documentation written for machine consumption, coding standards, and test contracts. AGENTS.md is the emerging open convention for this: a standard file format where a repository states build commands, conventions, and constraints that any coding agent (not just one vendor's) can read. Anthropic uses the parallel term "context engineering": deliberately deciding what information an agent sees, in what form, at what point in its task.

Why It Matters: Agents are only as good as the context and specification they're given; vague tickets and undocumented tribal knowledge produce plausible-looking but wrong code at agent speed, which is worse than the same failure mode at human speed because it's harder to catch. AGENTS.md has become a real cross-vendor standard in under two years: over 60,000 open-source repositories use it, and it's now stewarded by the Linux Foundation's Agentic AI Foundation with support from OpenAI, Google, GitHub, Cursor, and others (agents.md, 2026): meaning "does the repo have an AGENTS.md and is it accurate" is now a legitimate engineering-maturity signal you can ask about directly.

What I Need to Understand:

- This is a documentation and specification discipline, not a new AI capability: it requires the same rigor as writing a good design doc, and most orgs' documentation is not currently good enough to hand to an agent unsupervised.
- AGENTS.md (and equivalents like CLAUDE.md) typically encode: build/test commands, code style, directory conventions, "do not touch" areas, and security constraints: stale or wrong instructions here actively mislead agents rather than just failing to help.
- "Context engineering" is broader than one file: it includes what's in the agent's working memory, which tools it can call, and how much of the codebase it's shown, and getting this wrong is a common cause of agents producing code that's locally sensible but architecturally wrong.
- Specification quality becomes a bottleneck as agent capability increases: this is the emerging discipline sometimes called "spec-driven development," and it changes what "senior engineer" work looks like (writing specs and reviewing implementations vs. writing code).
- This is an investment with compounding returns: good repo-level context pays off across every future agent task in that codebase, unlike a one-off prompt.

Questions I Should Be Able to Ask My Team:

1. Do our repositories have agent instruction files (AGENTS.md, CLAUDE.md, or equivalent), who owns keeping them accurate, and when were they last verified against reality?
2. What happens when an agent's context is wrong or stale: do we have any detection for "agent acted on outdated instructions," or would we only find out from a bad PR?
3. Whose job is it now to write specifications precise enough for an agent to implement correctly, and is that a skill we're deliberately building in the team?

Technologies / Standards / Companies to Know: AGENTS.md (Linux Foundation Agentic AI Foundation), CLAUDE.md (Anthropic convention), .cursorrules (Cursor), Model Context Protocol (MCP) for tool/data access.

Recommended Learning:

- agents.md: the open standard: the spec itself and who backs it.
- Anthropic: Effective context engineering for AI agents: primary source on the discipline and its tradeoffs.
- Anthropic: Claude Code best practices: concrete guidance on repo instructions and agent workflows from the team building the tool.

Time Investment: 1 hour

4.4 AI Code Review, AI-Generated Testing & AI Debugging

Priority: Must Understand

Executive Definition: Three related but distinct capabilities: AI code review (a model reviews a diff and flags issues before or alongside human reviewers), AI-generated testing (agents write unit/integration tests, sometimes from specifications or from the code itself), and AI debugging (agents reproduce a bug, form a hypothesis, and propose or apply a fix). Each is now a standard offering from major platforms: GitHub reports 72.6% of developers using Copilot code review found it improved their effectiveness (GitHub Octoverse 2025): but each also introduces a specific new failure mode rather than simply removing manual effort.

Why It Matters: Agents now produce PR volume that outpaces human review capacity; DORA's 2025 research names this directly as the "velocity paradox": time saved writing code is consumed by verification, and reviewers face an asymmetric cognitive burden reviewing large AI-generated changesets (dora.dev, 2025). AI-generated tests can pass while testing the wrong thing (validating the implementation's behavior rather than the intended behavior), and AI debugging can "fix" a symptom without addressing the root cause: both failure modes look like success in a green CI pipeline.

What I Need to Understand:

- AI code review works best as a first-pass filter (style, obvious bugs, missing tests) that reduces human reviewer load, not as a replacement for a human's judgment on architecture, intent, or business logic: treat it as raising the floor, not replacing the ceiling.
- AI-generated tests are systematically vulnerable to a specific failure: if the same or a related model writes both the code and its tests, the tests can encode the bugs of the implementation rather than catching them: test provenance and independent review of test *intent* (not just coverage percentage) matters more than before.

- DORA's own recommendation is to shift AI feedback earlier (to the author while writing, not just the reviewer after the fact) and to deploy context-aware review agents for standards enforcement while reserving human review for judgment calls (dora.dev, 2025).
- GitClear's analysis of 211 million lines of code (2020-2024, including Google/Microsoft/Meta repos) found copy-pasted code roughly doubled as a share of changes (8.3% to 12.3%) while refactoring's share fell from 25% to under 10% (the first time duplicated code exceeded refactored code in this dataset (GitClear, 2025)) a concrete quality-erosion signal to watch for, independent of any vendor's review tool.
- Coverage metrics (percent of lines covered) become less meaningful when tests are AI-generated at volume; the question shifts to whether tests encode the actual specification/intent, which requires spot-checking, not just running the suite.

Questions I Should Be Able to Ask My Team:

1. Are our AI-generated tests being reviewed for whether they check the right behavior, or only whether they pass and raise coverage numbers?
2. What's our reviewer bottleneck right now: has PR volume grown faster than review capacity, and what specifically changed in our review process to compensate?
3. When AI debugging tools "fix" an issue, do we have any process that distinguishes a root-cause fix from a symptom patch, before it ships?

Technologies / Standards / Companies to Know: GitHub Copilot code review, CodeRabbit, Greptile, Graphite, Anthropic Claude (code review and debugging use), Sentry/observability-linked debugging agents.

Recommended Learning:

- DORA: Balancing AI tensions: the velocity paradox and concrete review-process recommendations, primary source.
- GitClear: AI Copilot Code Quality 2025 research: the copy-paste/refactoring data, based on 211M lines across major tech firms.
- GitHub Octoverse 2025: adoption data on Copilot code review (72.6% reporting improved effectiveness).

Time Investment: 2-3 hours

4.5 AI-Assisted Refactoring, Modernization & Technical Debt Reduction

Priority: Monitor

Executive Definition: Using AI (assistant, agent, or a purpose-built migration tool) to modernize legacy code (language/framework upgrades, dependency updates, dead code removal, and structural refactoring) at a scale and speed manual effort couldn't match. This is one of the better-evidenced

enterprise use cases: it's a bounded, verifiable task (behavior should be unchanged) rather than open-ended feature creation, which makes correctness easier to check automatically.

Why It Matters: This is where large, credible organizations report the clearest ROI. Google's engineering research group published a peer-reviewed account of an LLM-assisted migration tool used across 39 internal migration projects over twelve months: roughly 74% of the resulting code changes originated from the model, and developers reported roughly 50% time savings versus prior manual migrations, but with developers remaining firmly in the loop validating and guiding every change, not a hands-off pipeline (arXiv 2504.09691, Google, Foundations of Software Engineering (FSE) 2025). Amazon has separately publicized an AI-assisted Java modernization effort it says saved roughly 4,500 developer-years, a figure that should be treated as a vendor/leadership claim rather than an independently peer-reviewed result, but is directionally consistent with the Google data.

What I Need to Understand:

- The strongest evidence for AI-assisted modernization is in well-scoped, mechanically-checkable migrations (language version upgrades, API replacements, dependency bumps) with automated tests as a safety net: not in ambiguous "clean up this legacy system" mandates.
- Google's own reported model was developer-led automation, not autonomous refactoring: an algorithm found candidate locations, an LLM proposed changes, and developers validated each one: this is squarely "coding agent," not "software factory."
- Ironically, evidence from AI Code Review/Testing (GitClear) suggests day-to-day AI-assisted coding is currently *reducing* the proportion of refactoring work being done, even as purpose-built migration tooling shows strong results: the gain seems concentrated in dedicated modernization initiatives, not organic day-to-day cleanup.
- Test coverage is the precondition, not a side benefit: a codebase without strong automated tests cannot safely use AI for large-scale refactoring, because there's no automated way to confirm behavior didn't change.
- Vendor-published productivity claims (developer-years saved, percentage faster) generally lack independent verification or methodology detail: treat them as directional, and ask what was actually measured.

Questions I Should Be Able to Ask My Team:

1. Which parts of our tech debt backlog are mechanically-checkable migrations (good AI-refactoring candidates) versus ambiguous redesigns (poor candidates)?
2. What test coverage do we have on the systems we're proposing to AI-refactor, and is it sufficient to catch a behavioral regression?
3. If we adopt a migration tool or agent for this, what's the actual acceptance/validation process: who checks each change, and how much is genuinely automated versus human-reviewed?

Technologies / Standards / Companies to Know: Amazon Q Developer (transformation agents), GitHub Copilot (refactoring workflows), Google's internal LLM migration tooling (published research, not externally available), Anthropic Claude Code, OpenAI Codex: for large-scale migration, purpose-built

tooling generally outperforms generic chat assistants.

Recommended Learning:

- Migrating Code At Scale With LLMs At Google (arXiv 2504.09691, FSE 2025): the primary, peer-reviewed source with real numbers and methodology.
- InfoWorld: How Google is using LLMs for complex internal code migrations: accessible summary of the above paper.
- GitClear: AI Copilot Code Quality 2025 research: the counterpoint data on declining day-to-day refactoring share.

Time Investment: 1 hour

4.6 Parallel Coding Agents, Developer Supervision Ratios & Software Factories

Priority: Must Understand

Executive Definition: Rather than one developer working with one agent, some organizations are experimenting with one developer directing *several* agents working in parallel on different tasks: shifting the developer's role from implementer to supervisor/orchestrator. "Software factory" is the industry term for the more ambitious end-state: pipelines where issues flow to agents and PRs flow out with minimal human involvement. This is the least mature part of the continuum above: real vendor products exist, but independent, large-scale evidence of safe supervision ratios or factory-level throughput gains is thin as of late 2026.

Why It Matters: This is where the biggest headcount and org-design decisions get made prematurely on vendor claims rather than evidence: "one developer can now supervise N agents" is a governance and quality-assurance question, not a capability question, and the honest answer depends entirely on task type, codebase maturity, and review capacity, none of which vendors control for. DORA's "velocity paradox" already shows single-agent-per-developer review burden growing; multiplying the number of agents per developer without multiplying review/verification capacity increases risk faster than it increases output.

What I Need to Understand:

- There is no established, independently-validated "supervision ratio" (agents per developer): treat any specific number a vendor gives you as a marketing claim until you've seen it hold on a comparable codebase with your own quality bar.
- The binding constraint on parallel agents is almost never agent capability: it's the human's ability to review, integrate, and reconcile several concurrent streams of change without losing context on any of them.

- "Software factory" claims should be checked against the same rigor as any other late-stage continuum claim: what's the actual task class, what's the failure rate, and what human checkpoint exists before code reaches production.
- Parallelizing agents multiplies the downstream load on code review, CI, and architecture consistency: an org that hasn't solved single-agent review bottlenecks should not expect parallel agents to solve anything.
- This area is moving fast and current best practice will likely look dated within a year: treat specific product claims as time-stamped, and revisit the evidence base periodically rather than trusting a one-time briefing.

Questions I Should Be Able to Ask My Team:

1. If we run multiple agents per developer, what specifically increases to keep quality constant (review capacity, test automation, or something else) and have we actually provisioned that?
2. What's the evidence behind any specific "agents per developer" ratio a vendor or consultant has proposed to us: has it been demonstrated at our scale, or only in their marketing?
3. In a parallel-agent workflow, what happens when two agents produce conflicting changes to the same system, and who resolves that: is there a real process, or are we assuming it away?

Technologies / Standards / Companies to Know: Factory.ai, Devin (Cognition), GitHub Copilot workspace/multi-agent features, Anthropic's multi-agent orchestration patterns (published research, not a packaged product): this space consolidates quickly; expect vendor positioning to shift.

Recommended Learning:

- Anthropic: How we built our multi-agent research system: primary-source engineering account of orchestrator/subagent design tradeoffs (applicable beyond research use cases).
- DORA: Balancing AI tensions: the review-burden data that should temper supervision-ratio claims.
- Factory.ai: Factory 2.0: from coding agents to software factories: read as a vendor claims document to calibrate against, not as validated fact.

Time Investment: 1 hour

4.7 CI/CD, Quality Gates & Architecture Validation in an Agentic Development Model

Priority: Must Understand

Executive Definition: Traditional CI/CD (build, test, lint, deploy pipelines) was designed for human-paced change volume and human-authored code. Agentic development requires the same pipelines to do more: verify not just that code compiles and passes tests, but that it conforms to architectural constraints an agent has no inherent reason to respect (module boundaries, dependency rules, security policies).

"Architecture fitness functions" (automated, executable checks of architectural properties) are the emerging mechanism for this, extending a pre-AI concept (from evolutionary architecture practice) to constrain agent-generated code specifically.

Why It Matters: DORA is explicit that AI adoption has a *negative* relationship with software delivery stability, and that the reason is organizational, not technical: without strong automated testing, version control discipline, and fast feedback loops, increased change volume simply exposes existing weaknesses faster (dora.dev, 2025). CI/CD and architecture validation are the control systems that determine whether higher agent throughput translates into more working software or more incidents: this is the single highest-leverage investment for making the rest of this domain's topics safe.

What I Need to Understand:

- DORA's core 2025 finding on stability is not "AI breaks things": it's that AI removes the natural rate-limiting effect of human typing speed, so whatever gaps exist in your test automation and control systems get hit harder and faster (dora.dev, 2025).
- Quality gates need to check things beyond test pass/fail for agent-generated code specifically: dependency policy, architectural layering, security scanning, and license compliance: because an agent will happily generate code that passes tests while violating conventions no test encodes.
- "Fitness functions" (automated architectural checks, e.g., via ArchUnit-style tooling) are how teams encode architectural intent in a form agents and pipelines can actually enforce, rather than relying on a human reviewer to notice a violation.
- Small, frequent changes remain the safest unit of work even for agents: DORA specifically recommends enforcing small batch sizes as a countermeasure to unwieldy AI-generated changesets (dora.dev, 2025): "let the agent work for a long time and produce one huge PR" is a known anti-pattern.
- Platform engineering maturity is a leading indicator DORA ties directly to AI value realization (90% of high performers had strong platform engineering practices): CI/CD and quality-gate investment is not a side cost of AI adoption, it's the precondition for it paying off (dora.dev, 2025).

Questions I Should Be Able to Ask My Team:

1. What automated checks run against agent-generated PRs beyond unit tests (architectural rules, security scans, dependency policy) and what's not yet covered?
2. Has our delivery stability (change failure rate, Mean Time to Restore (MTTR)) moved since agent adoption, and do we have the DORA metrics instrumented to actually know?
3. Do we enforce small batch sizes for agent-generated changes, or are agents producing large changesets that overwhelm review and CI?

Technologies / Standards / Companies to Know: ArchUnit and similar fitness-function tooling, standard CI/CD platforms (GitHub Actions, GitLab CI, Jenkins) extended with agent-aware checks, DORA metrics (deployment frequency, lead time, change failure rate, MTTR) as the measurement backbone.

Recommended Learning:

- DORA: State of AI-assisted Software Development 2025: primary source on the stability finding and platform engineering correlation.
- DORA: Balancing AI tensions: concrete batch-size and quality-gate recommendations.
- InfoQ: Agentic fitness functions: extending evolutionary architecture beyond deterministic rules: how fitness-function practice is being adapted for agent-generated code.

Time Investment: 2-3 hours

4.8 AI-Generated Code Risk & Secure Development Practices

Priority: Must Understand

Executive Definition: AI-generated code carries security and quality risks distinct from human-authored code: models can introduce vulnerabilities even when explicitly asked to fix them, code volume can outpace security review capacity, and "vibe coding" (accepting AI output with minimal scrutiny) can bypass an organization's existing secure-development controls entirely. This topic is about the governance and technical controls needed so that increased AI-driven throughput doesn't translate into increased breach surface.

Why It Matters: DORA reports that even with 90% AI adoption, 30% of developers report little or no trust in AI-generated code (dora.dev, 2025): a meaningful trust gap at the point of production use. Peer-reviewed research on iterative AI code generation found vulnerabilities *increased* through refinement rather than decreased: a study of 400 code samples across 10 iterations each found critical vulnerabilities rose roughly 37.6% after five iterations, and notably, prompts explicitly requesting security improvements still introduced new vulnerabilities alongside the fixes (Institute of Electrical and Electronics Engineers (IEEE) International Symposium on Technology and Society (ISTAS) 2025 / arXiv 2506.11022). This directly contradicts the assumption that "just have the agent iterate more" improves safety.

What I Need to Understand:

- Security review cannot be an afterthought bolted onto an accelerated pipeline: the research shows iteration itself is not self-correcting for security, so static analysis, dependency scanning, and human security review need to run on every iteration, not just the final output.
- "Vibe coding" (a term popularized by Andrej Karpathy for accepting AI-generated code with minimal scrutiny) is a real and named enterprise governance concern; the Cloud Security Alliance and others have specifically flagged the gap between how individuals use these tools informally and what enterprise governance requires.
- Prompt intent doesn't reliably map to secure output: security-focused prompts in the referenced study still produced vulnerabilities (notably cryptographic errors), meaning "we told the agent to be secure" is not a control, it's a suggestion.

- GitClear's data on rising code duplication is also a security-relevant signal: duplicated code multiplies the surface area for a given vulnerability class rather than fixing it once in a shared location.
- The trust gap DORA measured (30% low/no trust) is itself a risk signal worth tracking over time: declining trust alongside rising adoption suggests teams are shipping code they don't fully believe in, which should trigger governance attention, not be dismissed as normal friction.

Questions I Should Be Able to Ask My Team:

1. What security scanning (Static Application Security Testing (SAST), dependency, secrets) runs specifically on AI-agent-generated PRs, and does it run per-iteration or only at final merge?
2. Do we have any policy distinguishing what an agent is allowed to touch unsupervised (e.g., auth, payments, data access layers) versus what always requires human security review?
3. What's our current measure of developer trust in AI-generated code, and if it's declining, what's driving that, and are we listening to it or overriding it with velocity targets?

Technologies / Standards / Companies to Know: Standard SAST/DAST and dependency-scanning tools (Snyk, Semgrep, GitHub Advanced Security) extended for agent-generated code volume; OWASP guidance on AI-assisted development; Cloud Security Alliance research on vibe-coding governance gaps.

Recommended Learning:

- Security Degradation in Iterative AI Code Generation (arXiv 2506.11022, IEEE ISTAS 2025): the peer-reviewed primary source on vulnerabilities increasing through iteration.
- DORA: State of AI-assisted Software Development 2025: the 30%-low-trust finding and broader risk framing.
- Cloud Security Alliance: The Vibe Coding Governance Gap: enterprise governance framing of the vibe-coding risk.

Time Investment: Half day

Future Engineering Workforce & Organizational Design

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/future-engineering-workforce-and-organizational-design/

5.1 Which Engineering Skills Appreciate vs Commoditize in the AI Era

Priority: Must Understand

Executive Definition: Some engineering skills are becoming faster and cheaper to produce with AI assistance (boilerplate code, routine CRUD, first-draft tests, syntax-level fluency in a given language): these are commoditizing. Others are becoming more valuable precisely because AI can't reliably do them (system architecture judgment, knowing what to build and why, verifying and debugging AI-generated code, security review, and managing ambiguity): these appreciate. The risk for leadership is conflating "AI writes code faster" with "we need fewer senior judgment-holders," when the evidence points the opposite direction.

Why It Matters: DORA's 2025 report found AI adoption has a positive relationship with throughput but a negative relationship with delivery stability unless the organization has strong existing engineering practices: meaning code volume is up, but so is the burden of review, verification, and architectural coherence, which are exactly the skills that don't commoditize. GitHub's 2025 Octoverse data shows AI tools are actively reshaping which languages and patterns developers reach for, which is a leading indicator that lower-level implementation choices are increasingly delegated to tooling while higher-level system decisions remain human.

What I Need to Understand:

- The DORA finding that trust in AI-generated code remains limited (roughly 30% of respondents report little or no trust in it) even as usage is nearly universal: a gap that puts a premium on review and verification skill, not a shortage of it
- Why "prompt engineering" as a standalone skill is already commoditizing (tools abstract it) while "context engineering" (structuring what information a model has access to, curating retrieval, managing agent memory and tool access) is emerging as a genuinely durable skill
- The distinction between a developer who ships AI-assisted code they can't explain and one who can explain, test, and defend it: only the second is doing work that appreciates
- That security review, architecture, and incident response skill demand is rising, not falling, as more code (including AI-authored code) enters production faster
- This is an active, contested area: there is no consensus taxonomy of "future-proof" skills, and confident lists of them should be read skeptically

Questions I Should Be Able to Ask My Team:

1. Of the code merged last month that was AI-assisted, what fraction went through the same review rigor as hand-written code, and can you show me a case where review caught something real?
2. Which of our senior engineers' skills would still be valuable if every line of new code were AI-generated tomorrow, and which of our juniors' skills would not be?
3. Are we hiring and promoting for "can prompt a model well" or for "can architect a system and verify what a model produced": and does our leveling rubric actually distinguish these?

Technologies / Standards / Companies to Know: GitHub Octoverse (annual developer ecosystem data), DORA State of AI-assisted Software Development, context engineering (emerging term, not yet standardized) as distinct from prompt engineering.

Recommended Learning:

- DORA 2025: State of AI-assisted Software Development: primary data on trust in AI code, throughput/stability tradeoffs, and what separates high performers.
- GitHub Octoverse 2025: annual primary data on how AI is changing developer behavior, language choice, and repo activity at scale.

Time Investment: 1 hour

5.2 The Junior Engineer Pipeline Problem

Priority: Must Understand

Executive Definition: AI coding assistants disproportionately automate the kind of well-specified, low-ambiguity work that junior engineers traditionally learn on: meaning fewer organizations are hiring at entry level, which breaks the pipeline that has always produced tomorrow's senior engineers. This isn't a hypothetical: it's a measurable early labor-market pattern, though its size and permanence are still being established.

Why It Matters: Stanford's Digital Economy Lab found the employment gap for workers aged 22-25 in AI-exposed occupations widened from 15% (July 2025) to 19% (June 2026) relative to less-exposed peers, driven primarily by reduced hiring rather than layoffs of existing staff: a pattern with direct relevance to any large engineering organization deciding whether to keep funding junior hiring. If your organization quietly stops hiring juniors because AI makes senior-heavy teams look more efficient today, you are consuming a pipeline you didn't build and have no plan to replace in five to ten years.

What I Need to Understand:

- The Stanford finding's own caveat: this is a documented correlation with a plausible mechanism, not proven causation: other factors (broader tech hiring slowdown, interest rates) are confounds the researchers themselves flag
- The mechanism specifically: reduced hiring, not increased firing: meaning the damage shows up in your pipeline plans quietly, not in a headline layoff you'd notice

- Why junior engineers historically learned architecture and judgment by doing the "boring" work AI now does, and what your organization's plan is to teach that judgment if the boring work disappears
- That this is a multi-year strategic risk (who becomes your senior engineers in 2032?) that a large enterprise engineering organization has more control over than commentary suggests: you can choose to keep hiring and training juniors even if the market average doesn't
- This remains an actively studied, evolving area: treat specific percentage figures as a snapshot, not a settled long-run trend

Questions I Should Be Able to Ask My Team:

1. How many junior/entry-level engineers have we hired in the last four quarters compared to two years ago, and was that a deliberate decision or a side effect of AI-driven "efficiency" targets?
2. If we stopped hiring juniors for three years, what would our senior engineering bench look like in 2031, and who has modeled that?
3. What is our actual plan for how a junior engineer develops architectural judgment when AI does the tasks they used to learn from?

Technologies / Standards / Companies to Know: Stanford Digital Economy Lab (Canaries dashboard: ongoing tracking, not a one-time study), IEEE Spectrum workforce reporting.

Recommended Learning:

- Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of AI (Stanford Digital Economy Lab): the primary research; read the caveats section, not just the headline stat.
- Canaries Dashboard (Stanford Digital Economy Lab): a living, updated data source rather than a static report, useful for tracking whether the gap is widening or stabilizing.

Time Investment: 1 hour

5.3 Changing Roles, Team Topology & Hiring/Competency Frameworks for the AI Era

Priority: Should Understand

Executive Definition: As AI agents take on more implementation work, the shape of engineering teams and the definition of roles like staff engineer and engineering manager are shifting: staff engineers increasingly spend time directing and reviewing agent output rather than only writing code themselves, and managers are starting to think about "how many agents can one engineer effectively supervise" the way they once thought about span of control over people. None of this has settled into an industry-standard org chart or competency framework yet: it is actively being worked out in public by practitioners.

Why It Matters: Getting team topology wrong here has a direct cost: DORA's 2025 report identifies seven distinct team archetypes ranging from "foundational challenges" (low performance, high burnout) to "harmonious high achievers," and which archetype a team falls into is strongly related to whether roles and workflows were deliberately redesigned for AI-assisted work or simply had AI tools bolted onto an unchanged structure. Your hiring rubrics and leveling guides (written for a world where seniority mostly meant "writes more/better code") need updating for a world where reviewing, directing, and verifying AI output is itself a core skill being evaluated.

What I Need to Understand:

- There is no settled "engineer-to-agent ratio": any specific number offered as a benchmark right now is speculative, not empirical, and should be treated that way in this document and in vendor pitches
- Team Topologies' existing vocabulary (stream-aligned, platform, enabling, complicated-subsystem teams) is being actively extended by its own authors to describe agent-inclusive teams: this is the most credible existing framework to anchor on, not a from-scratch invention
- How staff engineer and EM job descriptions are changing in practice: more time on review/verification/architecture, less time on raw output, and whether your leveling criteria and interview loops have caught up
- The risk of over-rotating: treating "can direct AI agents" as a proxy for seniority without also verifying the underlying engineering judgment that makes that direction sound
- This is one of the least mature topics in this document: flag any confident claim about "the right ratio" or "the new org chart" as an opinion, not a finding

Questions I Should Be Able to Ask My Team:

1. Have we actually updated our staff engineer and EM leveling criteria and interview loops for AI-assisted work, or are we evaluating people against a pre-AI rubric while expecting AI-era output?
2. When an engineer directs multiple AI agents at once, who is accountable when something goes wrong in production, and does our incident process reflect that?
3. Where did we get any "agents per engineer" ratio we're using for planning, and would we accept that level of evidence for any other headcount decision?

Technologies / Standards / Companies to Know: Team Topologies (Matthew Skelton & Manuel Pais: the org design framework being actively extended for AI), DORA team archetypes, GitHub/Anthropic/OpenAI internal engineering blogs (useful as case studies, not standards).

Recommended Learning:

- Team Topologies as the "infrastructure for agency" with AI: the framework's own authors extending it for agent-inclusive teams; primary and current.
- DORA 2025: State of AI-assisted Software Development: the seven team archetypes and what separates high performers from teams in "foundational challenges" mode.

Time Investment: 1 hour

AI for Professional & Knowledge Workers

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/ai-for-professional-and-knowledge-workers/

6.1 The Maturity Ladder: Chat → Assistants → Connected Assistants → Workflows → Agents → Autonomous Business Processes

Priority: Must Understand

Executive Definition: Six rungs describe how AI moves from novelty to operational infrastructure: **Chat** (ad hoc prompting, no system access), **Assistants** (embedded in one app: Word, Excel), **Connected Assistants** (cross-system retrieval with permissions, e.g., enterprise search), **Workflows** (deterministic multi-step processes with an AI step inserted), **Agents** (goal-directed, tool-using, makes some decisions independently), and **Autonomous Business Processes** (agents run an end-to-end process with human oversight by exception, not by step). Most enterprise deployments today sit at rungs one through three; vendor marketing routinely implies rungs five and six.

Why It Matters: Gartner calls the gap between marketed and actual autonomy "agentwashing" (most products sold as "agents" are still reactive assistants) and separately predicts over 40% of agentic AI projects will be canceled by end of 2027 due to unclear ROI, escalating cost, and weak risk controls (Gartner). MIT's Networked Agents and Decentralized AI (NANDA) initiative found 95% of enterprise GenAI pilots produced no measurable return, and traced the divide not to model quality but to whether organizations redesigned the workflow around the tool or just added a chat box on top of an unchanged process (MIT NANDA). This ladder is the vocabulary for scoping a project honestly and refusing inflated claims.

What I Need to Understand:

- Each rung requires different governance: approval-per-action at "Workflows," approval-by-exception at "Autonomous Business Processes": the risk model changes at every step, not just the tech.
- Moving up a rung is a process-redesign exercise, not a model swap or a licensing upgrade.
- "Adding chat" to an existing workflow is a rung-one or rung-two change even when marketed as agentic.
- McKinsey found high performers were three times more likely to have "fundamentally redesigned workflows" than typical adopters (74% vs. 25%), and that this (not tool adoption) is what correlates with financial impact (McKinsey).
- Vendor claims of "agent" should be interrogated against Gartner's stage definitions before budget commitment.

Questions I Should Be Able to Ask My Team:

1. Which rung is each of our current AI deployments actually operating at, and who verified that classification independent of the vendor's own description?
2. For anything sold to us as "agentic," what decision is the system actually making autonomously, versus what is still a scripted step with an LLM call inserted?
3. What changed in the underlying business process (not just the tooling) for our highest-value AI deployment, and can we show a before/after process map?

Technologies / Standards / Companies to Know: Gartner Hype Cycle for Agentic AI, MIT NANDA "State of AI in Business," Microsoft Agent 365, Forrester Adaptive Process Orchestration.

Recommended Learning:

- Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027: cost, ROI, and governance reasons projects fail.
- Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026: lays out the five-stage maturity progression through 2029.
- MIT NANDA Project Overview: source of the "GenAI Divide" pilot-failure research (note: methodology has drawn some independent criticism over sample size; treat findings as directionally useful, not definitive).
- McKinsey: The State of AI: workflow-redesign correlation with financial impact.

Time Investment: 2-3 hours

6.2 Enterprise AI Assistants & Enterprise Search

Priority: Must Understand

Executive Definition: Two distinct product categories get conflated under "AI assistant." App-embedded copilots (Microsoft 365 Copilot in Word/Excel, Gemini in Docs/Sheets) work inside one application. Enterprise search/assistant platforms (Glean, Google Gemini Enterprise, Copilot with Microsoft Graph connectors, Claude Enterprise with connectors) index content across many systems and answer questions grounded in whatever the requesting user is already permitted to see. The second category's value and risk both hinge on one mechanism: permission-aware retrieval: not model quality.

Why It Matters: Glean's documentation is explicit that answers are only as good as the underlying permissions model: the system builds a knowledge graph from connector-crawled content, activity, and identity data, and filters every result through existing access controls before it reaches the model (Glean). Get this wrong and you either expose data across permission boundaries or produce thin, unhelpful answers because whole systems weren't indexed: both are governance failures dressed up as AI-quality complaints.

What I Need to Understand:

- App-embedded copilots and cross-system enterprise search solve different problems and are frequently bought and evaluated as if interchangeable.

- Connector/indexing coverage (which systems are actually crawled) determines usefulness far more than which model powers the assistant.
- Permission propagation lag (offboarding, role changes, revoked access) is a real security gap if the AI's index doesn't sync in near-real time with source-system ACLs (Glean).
- "Hallucination" complaints in an enterprise-search context are often retrieval/grounding failures (wrong or missing source document), not model failures: the fix differs accordingly.
- Licensing is typically per-seat and stacks on top of, not instead of, existing SaaS licenses: check for overlap before buying a second search layer.

Questions I Should Be Able to Ask My Team:

1. What percentage of our critical knowledge systems are actually indexed by this assistant, and what's explicitly excluded?
2. When someone's access is revoked or changes, how long until that's reflected in what the AI can retrieve and answer from?
3. Can we trace any given answer back to the specific documents and the permission check that allowed them to be used?

Technologies / Standards / Companies to Know: Microsoft 365 Copilot + Graph connectors, Google Gemini Enterprise (formerly Agentspace), Glean, Claude Enterprise + Anthropic connectors/MCP.

Recommended Learning:

- Glean Search FAQ: how permissions-aware indexing and grounding actually work.
- Glean Developer: Permissions: the technical model for access control in indexing.
- Claude Enterprise: identity, audit, and data-retention controls to expect from this category.
- Microsoft Agent 365 Overview: Microsoft's emerging identity/governance layer for agents and connected assistants.

Time Investment: 2-3 hours

6.3 Deep Research Agents & Document/Presentation/Spreadsheet Generation

Priority: Monitor

Executive Definition: Deep research agents (OpenAI Deep Research, Google Gemini Deep Research, NotebookLM Deep Research, Microsoft Researcher in Copilot Notebooks) run asynchronously for minutes rather than seconds: they plan a research strategy, search and read across many sources, and return a cited report: closer to delegating to a junior analyst than to prompting a chatbot. These increasingly connect directly to document generation, turning that output into a Word doc, slide deck, or spreadsheet.

Why It Matters: OpenAI's own account of Deep Research describes it finding, analyzing, and synthesizing "hundreds of online sources" into an analyst-grade report in 5-30 minutes, built on a reasoning model tuned specifically for multi-step browsing tasks (OpenAI). The genuine capability shift is real, but benchmark scores (GAIA, Humanity's Last Exam) measure general research aptitude, not accuracy on your specific domain, and every generated report still needs a human citation-verification pass before external use.

What I Need to Understand:

- The asynchronous, multi-minute execution model changes how this fits into a workflow: it's not a replacement for instant chat, it's a delegated task.
- Grounding source matters: public-web research (OpenAI, most consumer deep research) versus enterprise-data-grounded research (Gemini Deep Research pulling from indexed internal sources) are different risk and value propositions (Google Cloud).
- Citations must be independently verified before a report leaves the building: misattribution and fabricated sources remain a known failure mode of these systems.
- This is distinct from templated document automation (mail merge, standard reporting), which is deterministic, cheaper, and should not be replaced by an agent for the sake of it.
- "High steerability" (custom tone, structure, format) is a real differentiator between vendors and worth testing on your own use cases, not vendor demos.

Questions I Should Be Able to Ask My Team:

1. When this tool cites a source, what's our process for verifying that citation before the output is used in an external-facing document?
2. Is this agent grounded in public web data, our own enterprise data, or both, and does that vary by user or by query?
3. What's the measured time and cost per report on our own use cases, compared to an analyst doing the same task: not the vendor's benchmark numbers?

Technologies / Standards / Companies to Know: OpenAI Deep Research, Google Gemini Deep Research / NotebookLM, Microsoft Researcher (Copilot Notebooks), Anthropic Claude web search and citations.

Recommended Learning:

- OpenAI: Introducing Deep Research: how the agent plans, browses, and synthesizes; benchmark results.
- Google Cloud: Use the Gemini Deep Research Agent: enterprise-grounded research workflow and use cases.
- Microsoft Support: Use Researcher in Copilot Notebooks: Microsoft's equivalent inside the Microsoft 365 (M365) stack.
- Google: NotebookLM Adds Deep Research: source-grounded research over uploaded documents.

Time Investment: 1 hour

6.4 Meeting Intelligence, Departmental Digital Workers & Function-Specific Agents (HR, Finance, Legal, Procurement, Sales, Operations, Program Management)

Priority: Should Understand

Executive Definition: Two related but distinct categories. Meeting intelligence tools (Otter, Microsoft Copilot in Teams, Google Gemini in Meet, Read.ai, Fireflies) transcribe, summarize, and extract action items from calls. Departmental "digital worker" agents are embedded directly in a system of record and scoped to its data: Salesforce Agentforce (sales/service), SAP Joule (finance, HR, supply chain), Workday agents (HR), ServiceNow agents (IT/operations). Because they're bounded to one platform's data model, these are typically the most concretely deployed and measurable category of enterprise AI today.

Why It Matters: SAP has shipped roughly 15 named Joule agents across finance, HR, and supply chain functions (Techzine; independent commentary from HR analyst Josh Bersin covers the same rollout: Bersin), and McKinsey found large enterprises (>$1B revenue) scaling AI agents at 40%, versus 22% flat for smaller organizations: the gap that matters for a mid-size enterprise sizing its own ambitions realistically (McKinsey). The recurring trap is the same as elsewhere: a meeting summarizer or departmental agent that reproduces a report you already generate is not transformation: it only counts if the underlying process changed.

What I Need to Understand:

- Meeting intelligence tools introduce their own governance surface: recording consent, retention policy, and who can query historical meeting transcripts across the org.
- Departmental agents are generally vendor-locked to that platform's data model: cross-platform orchestration (e.g., a Workday agent triggering a SAP process) is immature and should not be assumed.
- Pricing for departmental agents is increasingly per-agent or per-outcome/conversation rather than per-seat, which is harder to forecast and budget than traditional SaaS licensing.
- Function heads (HR, Legal, Procurement) will be sold these agents directly by vendors, often bypassing central IT/architecture review: a coordination point is needed before contracts land.
- Maturity varies sharply by function: sales/service (Agentforce) and IT operations (ServiceNow) are furthest along; legal and procurement agent tooling is comparatively immature and less independently validated.

Questions I Should Be Able to Ask My Team:

1. Who owns governance for meeting recordings and transcripts: retention period, consent, and who can search past meetings org-wide?
2. For each departmental agent under evaluation, what specific manual process does it replace, and can we see the actual before/after workflow, not just the vendor demo?
3. How does this agent's pricing scale if adoption succeeds and usage triples: is that cost modeled anywhere yet?

Technologies / Standards / Companies to Know: Microsoft Copilot in Teams, Google Gemini in Meet, Otter.ai, Read.ai, Salesforce Agentforce, SAP Joule, Workday agents, ServiceNow AI agents.

Recommended Learning:

- Microsoft: Catch Up on Meetings with Copilot in Teams: what meeting-recap AI actually does and doesn't capture.
- Salesforce: Agentforce Developer Guide: Get Started: primary technical documentation, not marketing.
- Josh Bersin: SAP Jumps Ahead in AI Agents with Joule: independent analyst view on the HR/finance agent rollout.
- Deloitte: The State of AI in the Enterprise: cross-functional adoption data.

Time Investment: 1 hour

Workflow Automation

Last updated September 11, 2026 · 4 min read · https://erikcaldwell.com/field-guide/workflow-automation/

7.1 Robotic Process Automation (RPA) vs Business Process Management (BPM) vs API Automation vs Agentic Workflows

Priority: Must Understand

Executive Definition: Four approaches to automating work, distinguished by how much of the control flow is deterministic versus probabilistic. **RPA** drives existing UIs like a human would (clicks, keystrokes). **BPM** orchestrates multi-step processes (including human approval steps and SLAs) against an explicit process model. **API/integration automation** (iPaaS) connects systems directly at the data layer. **Agentic workflows** use an LLM to plan and choose the next action dynamically, and can call RPA, BPM, or API steps as tools rather than replacing them outright.

Why It Matters: UiPath's own documentation frames the distinction cleanly: robots are "deterministic, rule-based," while agents are "probabilistic, adaptive" (UiPath): that's the axis that actually matters for architecture decisions, not "old technology vs. new technology." Forrester has named a new category, Adaptive Process Orchestration, specifically because it judges RPA and BPM platforms as "not architecturally designed" to support autonomous, nondeterministic decision-making at the same time as traditional workflow logic (Forrester).

What I Need to Understand:

- The real distinction is deterministic vs. probabilistic control flow: everything else (UI-level vs. API-level vs. process-level) is a secondary axis.
- RPA is fragile to interface changes because it operates at the UI layer; API automation is more stable but requires actual API access to the target system.
- BPM adds explicit process modeling and human-in-the-loop steps that RPA and pure API automation don't provide on their own.
- Agentic workflows are compositional, not a replacement category: a well-designed agent calls RPA/API/BPM steps as tools rather than reinventing them.
- The failure-mode and risk profile differ sharply by category: a deterministic bot fails loudly and traceably; a probabilistic agent can make a wrong autonomous decision silently in a regulated process (finance, HR, legal): governance must match.

Questions I Should Be Able to Ask My Team:

1. For this specific process, is the decision logic actually well-defined enough to automate deterministically, or are we reaching for an agent because mapping the real rules is harder than it looks?

2. When the agent's action is wrong, what's the human checkpoint, the reversal mechanism, and the audit trail: does it match what our existing RPA/BPM tools already provide?
3. Which of our current RPA, BPM, or API automation investments would an agentic layer actually replace, versus simply call as a tool underneath it?

Technologies / Standards / Companies to Know: UiPath, Microsoft Power Platform, Workato, Appian, Pega; Forrester's "Adaptive Process Orchestration" category.

Recommended Learning:

- Forrester: Announcing the Evaluation of the Adaptive Process Orchestration Market: the analyst-defined category bridging RPA/BPM and agentic automation.
- UiPath Docs: About Agents: vendor's own deterministic-vs-probabilistic framing, worth reading critically.
- Workato: Agentic (docs): how an integration-first (iPaaS) platform is layering agent orchestration onto deterministic automation.
- Microsoft Learn: Overview of Process Mining in Power Automate: how you discover the actual deterministic process before automating it.

Time Investment: 2-3 hours

7.2 Automation Platform Landscape & Where Deterministic Automation Still Wins (Microsoft Power Automate, UiPath, Workato, n8n, Zapier)

Priority: Should Understand

Executive Definition: Microsoft Power Automate (native to the Microsoft/Power Platform stack, licensing often bundled with M365), UiPath (enterprise RPA incumbent, repositioning around agentic orchestration), Workato (enterprise integration/iPaaS-first, adding agent orchestration on top), n8n (open-source, self-hostable, developer-oriented, node-based), and Zapier (SMB-oriented, largest app-connector catalog, lightweight setup) represent different points on cost, control, and openness: not a single ranked list. Deterministic automation on any of these remains the correct choice for high-volume, well-specified, compliance-sensitive work; agentic layers add cost and non-determinism that should be reserved for genuinely unstructured tasks.

Why It Matters: Every vendor in this space is now marketing "agentic" capability, which is exactly the "agentwashing" Gartner warns about (see Maturity Ladder topic above): the platform landscape hasn't changed as much as the branding has. The underlying economics still favor deterministic execution at volume: rule-based steps cost a fixed, predictable amount per transaction, while each LLM-driven decision in an agentic step adds inference cost, latency, and retry risk that scales with usage rather than staying flat.

What I Need to Understand:

- Platform choice is frequently already constrained by existing stack: Microsoft shop → Power Automate; heavy SaaS-to-SaaS integration → Workato/iPaaS; engineering/ops-owned automation → n8n.
- Deterministic automation is dramatically cheaper per transaction at scale than agent-driven automation: token/inference cost accrues per decision; rule-based execution doesn't.
- Compliance and audit requirements (SOX, financial controls, regulated processes) often mandate reproducible, traceable logic that agentic steps complicate as audit evidence.
- Open-source/self-hosted (n8n) versus SaaS (Zapier, Workato, Power Automate) shifts total cost of ownership from license fees to internal engineering maintenance time: a real tradeoff, not a free lunch.
- Vendor lock-in risk differs by platform: proprietary connector catalogs (Zapier, Workato) versus more portable, self-hosted workflow definitions (n8n).

Questions I Should Be Able to Ask My Team:

1. For a given automated process, what's the actual measured cost-per-transaction of an agentic approach (inference cost, retries, human review) versus the deterministic path we already have running?
2. Which of these platforms are we already paying for through existing licensing (M365, other SaaS bundles), and are we about to buy a second tool that duplicates that capability?
3. When an "agentic" step in a workflow makes a decision, what's our audit trail if we later need to explain or reverse that decision to a regulator or auditor?

Technologies / Standards / Companies to Know: Microsoft Power Automate, UiPath, Workato, n8n, Zapier, Appian.

Recommended Learning:

- Microsoft Learn: Overview of Process Mining in Power Automate: discovering deterministic process reality before choosing a platform.
- UiPath: What is Agentic Orchestration?: the RPA incumbent's own account of layering agents onto existing bots.
- Workato Docs: Agentic: an iPaaS vendor's technical documentation of agent orchestration on top of deterministic integration.
- n8n Docs: primary documentation for the open-source, self-hostable option.

Time Investment: 1 hour

Enterprise Knowledge & Data Architecture

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/enterprise-knowledge-and-data-architecture/

8.1 Retrieval-Augmented Generation (RAG), Embeddings & Vector Databases

Priority: Must Understand

Executive Definition: Retrieval-augmented generation (RAG) grounds a model's answer by first pulling relevant passages from your own data, rather than relying only on what the model learned in training. Text is converted into embeddings (numeric vectors capturing meaning) and stored in a vector database that finds semantically similar content at query time. This is the standard way enterprises connect a general-purpose model to proprietary knowledge without retraining it.

Why It Matters: RAG output quality is now understood to be dominated by retrieval and data quality, not model choice: Anthropic found that adding generated context to chunks before embedding ("contextual retrieval") cut retrieval failure rates by up to 49%, and combining it with reranking cut failures by 67% versus baseline vector search. Most "hallucination" complaints in production RAG systems are retrieval or chunking failures, not model failures: which is why this is the foundational topic for the entire domain.

What I Need to Understand:

- Embeddings are lossy numeric compressions of meaning; retrieval quality depends on chunking strategy, embedding model choice, and index freshness: not on the LLM.
- Vector databases (pgvector, Pinecone, Weaviate, etc.) are storage/retrieval infrastructure, not reasoning engines; choosing one is an infrastructure decision, not an AI strategy decision.
- Naive "chunk and embed" RAG degrades on long, structured, or technical documents because chunking destroys context: this is why contextual retrieval and hybrid retrieval (next topic) exist.
- Reranking (a second, more expensive relevance pass over retrieved candidates) materially improves precision and is usually worth the added latency/cost for high-stakes use cases.
- Retrieval accuracy and generation accuracy should be measured and evaluated separately; a wrong answer is often a retrieval-stage failure, not a model-stage one.

Questions I Should Be Able to Ask My Team:

1. How do we measure retrieval recall/precision separately from final answer quality, and what's our current baseline?
2. Are we using contextual retrieval and/or reranking, and what lift did we actually measure versus plain vector search?

3. What's our re-indexing cadence for documents that change frequently, and how stale can the index get before it's a risk?

Technologies / Standards / Companies to Know: pgvector, Pinecone, Weaviate, Qdrant, Azure AI Search, Amazon Kendra/OpenSearch, OpenAI/Cohere/Voyage embedding models, Anthropic Contextual Retrieval, Cohere Rerank.

Recommended Learning:

- Contextual Retrieval (Anthropic): the primary source for the retrieval-failure-rate numbers above.
- Retrieval-Augmented Generation for Knowledge-Intensive Natural Language Processing (NLP) Tasks (Lewis et al., 2020): the original RAG paper; establishes the pattern's fundamentals.
- pgvector (GitHub): illustrative of what a vector database actually is at the engineering level.

Time Investment: 2-3 hours

8.2 Hybrid Retrieval, Knowledge Graphs, GraphRAG & Text-to-SQL

Priority: Monitor

Executive Definition: Pure vector search retrieves by semantic similarity but is blind to exact keywords, IDs, and relationships between entities. Hybrid retrieval combines vector search with keyword search (e.g., Best Matching 25 (BM25)); knowledge graphs and GraphRAG structure information as entities and relationships so a system can reason across an entire corpus, not just within single documents; text-to-SQL lets a model query structured databases in natural language instead of treating tables as unstructured text.

Why It Matters: Many enterprise questions ("who approved this vendor, and what else are they connected to") are relationship or corpus-wide questions that plain vector RAG answers poorly: Microsoft Research built GraphRAG specifically because standard RAG cannot answer "global" queries requiring synthesis across many documents. Text-to-SQL benchmark accuracy also drops sharply on real, messy production schemas compared to curated academic benchmarks, so vendor accuracy claims need validation against your own data before you trust them.

What I Need to Understand:

- Hybrid search (vector + keyword) is close to default practice now, because pure semantic search misses exact matches like IDs, names, and codes.
- GraphRAG builds a knowledge graph and community summaries at indexing time: it costs meaningfully more to build and maintain than plain vector RAG and is justified only for corpus-wide relationship reasoning, not general document Q&A.
- Text-to-SQL accuracy on real enterprise schemas (ambiguous columns, undocumented joins) is typically much lower than published benchmark numbers: test on your own schema before trusting it.

- These techniques are additive: production systems commonly combine vector, keyword, and structured (SQL/graph) retrieval behind one interface rather than choosing just one.

Questions I Should Be Able to Ask My Team:

1. Which of our use cases genuinely require corpus-wide relationship reasoning (GraphRAG-shaped) versus simple document lookup (RAG-shaped)?
2. What is our text-to-SQL accuracy on our actual production schema, and what's the fallback when a generated query is wrong: silent execution or confirm-before-run?
3. What's the ongoing cost of keeping a knowledge graph or hybrid index synchronized with source systems as they change?

Technologies / Standards / Companies to Know: Microsoft GraphRAG, Neo4j, Elasticsearch/OpenSearch (BM25 + vector), Weaviate hybrid search, the BIRD-SQL (Big Bench for Large-Scale Database Grounded Text-to-SQL Evaluation) and Spider benchmarks.

Recommended Learning:

- GraphRAG: Unlocking LLM discovery on narrative private data (Microsoft Research): why standard RAG fails on corpus-wide queries.
- Microsoft GraphRAG project page: the reference implementation and architecture.
- BIRD-SQL benchmark: a realistic (not toy) text-to-SQL benchmark, useful as a sanity check for vendor claims.
- Hybrid search (Weaviate Documentation): technical explanation of combining BM25 and vector scoring.

Time Investment: 1 hour

8.3 Permissions-Aware Retrieval & Access Control Propagation

Priority: Must Understand

Executive Definition: When an AI system retrieves from enterprise content, it must enforce the same access permissions the source systems already have: who can see which documents, folders, or records. Permissions-aware retrieval means access decisions are checked at query time against retrieved content, not just at the login screen of the application in front of the model.

Why It Matters: This is one of the most common and dangerous production failures in enterprise RAG: a vector index built once from "all documents" and queried by everyone, regardless of source permissions, turns a chatbot into a data exposure path, as OWASP's RAG security guidance details explicitly. Standard identity and access management (Role-Based Access Control (RBAC)/Attribute-Based Access Control (ABAC)) built for applications does not automatically propagate into a vector database or embeddings pipeline: it has to be deliberately reimplemented at the retrieval layer, and getting it wrong is a governance and legal exposure issue, not a minor bug.

What I Need to Understand:

- A vector index has no inherent concept of "who can see what" unless permissions metadata is attached to every chunk and enforced at retrieval time (pre-filtering, not just post-filtering).
- Filtering after retrieval is weaker than filtering during retrieval: ranking on content a user shouldn't see can leak information even if the final response is blocked.
- Permission changes (offboarding, role changes, reclassification) must propagate into the retrieval layer promptly; sync lag is itself a risk window.
- Document-level and row-level security at the database/vector-store layer is different from, and complementary to, application-layer authorization: both are needed for defense in depth.
- Multi-tenant AI deployments (one system serving many business units or customers) need real namespace/partition isolation in the vector store, not just query-time filters.

Questions I Should Be Able to Ask My Team:

1. Does our retrieval layer enforce source-system permissions per query, or was the index built once from a static export at a single access level?
2. What is the sync lag between a permission change in the source system and that change taking effect in retrieval?
3. Have we actually red-teamed whether a user can retrieve content they shouldn't see through prompt-based probing, not just through the normal UI?

Technologies / Standards / Companies to Know: OWASP LLM/RAG security guidance, RBAC/ABAC, Postgres row-level security, Pinecone namespaces, Weaviate multi-tenancy, Microsoft Purview, Amazon Kendra/Q with ACL-aware retrieval.

Recommended Learning:

- RAG Security Cheat Sheet (OWASP): the primary reference for RAG-specific access-control risks.
- RAG with Permissions (Supabase Docs): a concrete, technical illustration of enforcing row-level permissions at the retrieval layer.

Time Investment: 1 hour

8.4 Data Classification, Residency, Freshness & Source Provenance

Priority: Must Understand

Executive Definition: Before content reaches a model, it needs metadata: sensitivity classification (public/internal/confidential/regulated), residency (where it may legally be stored/processed), freshness (how current it is), and provenance (source system, author, version). Without this metadata layer, an AI system cannot reliably enforce policy, explain its answers, or avoid presenting outdated information as current.

Why It Matters: Regulated environments increasingly require AI outputs to be traceable to a specific, current, authorized source: an answer with no provenance is an audit and liability problem, not just a UX gap. Data residency requirements constrain which cloud regions and model providers are legally usable for a given data set, which makes this a governance-driven architecture constraint rather than an engineering preference; NIST's AI Risk Management Framework explicitly calls for documenting data provenance and quality as part of AI governance (Govern/Map functions), signaling this is becoming a compliance expectation rather than optional hygiene.

What I Need to Understand:

- Classification, residency, freshness, and provenance are metadata problems solved in the data layer before retrieval: they cannot be bolted onto the model afterward.
- Residency decisions determine which cloud regions and model providers are usable for a given data set, end-to-end: including where embeddings are computed and cached, not just where the final answer is generated.
- "Freshness" needs an explicit owner and update cadence per source; an index with no last-verified date is a silent risk.
- Provenance (source system, author, timestamp, version) should be attached to every retrieved chunk and, ideally, surfaced as a citation in the answer.

Questions I Should Be Able to Ask My Team:

1. Can we trace any AI-generated answer back to the specific source document, version, and last-verified date it came from?
2. Which of our data sources have residency constraints, and does our current retrieval/model deployment actually satisfy them end-to-end?
3. Who owns freshness for each major knowledge source, and what's the process when a source becomes stale or deprecated?

Technologies / Standards / Companies to Know: NIST AI Risk Management Framework, ISO/IEC 42001, data catalogs (Microsoft Purview, Collibra, Atlan), cloud region/residency controls (Azure, AWS, GCP).

Recommended Learning:

- AI Risk Management Framework (NIST): the governance reference underlying provenance/classification obligations.
- AI RMF: Generative Artificial Intelligence Profile (NIST): GenAI-specific guidance on data provenance and quality.

Time Investment: 1 hour

8.5 Enterprise, Personal & Agent Memory Architecture

Priority: Should Understand

Executive Definition: "Memory" refers to information retained across sessions rather than supplied fresh each time: enterprise memory (shared organizational knowledge), personal memory (per-user preferences/history), and agent memory (an agent's record of what it has done in a task). Unlike RAG, which retrieves from a fixed corpus, memory is written and updated by the system itself over time: which raises its own accuracy, staleness, and governance questions.

Why It Matters: Memory expands what an agent can do across long-running or multi-session tasks, but introduces a distinct risk: the system now writes and later trusts its own prior conclusions, so errors can compound: which is why Anthropic's agent-memory approach treats memory as something to curate deliberately, not log unconditionally. Memory design overlaps directly with the permissions and provenance questions above: whose memory is it, who can read it, and does it expire?

What I Need to Understand:

- Memory is not the same as context window size (see Context Windows topic): it's about what persists after a task ends, versus what fits in one request.
- Agent memory architectures (e.g., the MemGPT "OS-inspired" model) keep a small working context plus a larger external store the agent retrieves from: this is retrieval applied to an agent's own history, subject to the same quality risks as RAG.
- Enterprise/shared memory needs the same permission and provenance controls as any other knowledge source: memory is a data asset, not just a convenience feature.
- Unreviewed agent-written memory can drift or encode errors that reinforce over time; production systems need review/expiry mechanisms, not indefinite accumulation.

Questions I Should Be Able to Ask My Team:

1. When an agent "remembers" something across sessions, who can see that memory, and does it inherit the access controls of the source data it was derived from?
2. What's our process for reviewing, correcting, or expiring agent-written memory so errors don't compound?
3. Is personal/user memory scoped and deletable in line with our data retention and privacy obligations?

Technologies / Standards / Companies to Know: Anthropic memory tool / Claude Managed Agents memory, MemGPT, Letta, Mem0.

Recommended Learning:

- Memory for Claude Managed Agents (Anthropic): a production example of curated (not unconditional) agent memory.
- MemGPT: Towards LLMs as Operating Systems (arXiv): the foundational paper behind most current agent-memory architectures.

Time Investment: 1 hour

Foundation Models & AI Engineering

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/foundation-models-and-ai-engineering/

9.1 Frontier, Reasoning & Multimodal Models

Priority: Must Understand

Executive Definition: Frontier models are the current highest-capability general-purpose models from the major labs (Anthropic Claude, OpenAI Generative Pre-trained Transformer (GPT), Google Gemini). "Reasoning" models are variants trained or configured to spend extra inference-time computation working through a problem step by step before answering, trading latency and cost for accuracy on harder tasks. Multimodal models accept and/or produce more than text (images, audio, video, and documents) within the same model.

Why It Matters: The three labs now ship closely competitive, frequently-updated models rather than one stable leader: Anthropic's Claude Opus 4.5, OpenAI's GPT-5.1, and Google's Gemini 3 all shipped within months of each other in late 2025, so architecture decisions should assume ongoing model churn rather than a fixed, permanent choice. Reasoning mode is a cost/latency dial, not a universal upgrade, and should be applied selectively rather than defaulted on everywhere.

What I Need to Understand:

- Extended reasoning ("thinking") trades latency and token cost for accuracy on complex tasks: enable it selectively for tasks that need it, not as a blanket default.
- Frontier leadership rotates every few months across labs; procurement and architecture should assume multi-model flexibility (see Model Routing topic) rather than long-term single-model commitment.
- Multimodal input (documents, images, audio, video) opens real new use cases, but quality varies significantly by modality and provider: validate per use case rather than trusting headline benchmarks.
- Benchmark leaderboards are frequently gamed or non-representative of your task mix: insist on evaluation against your own representative tasks before switching models.

Questions I Should Be Able to Ask My Team:

1. Which of our workloads actually need reasoning/extended-thinking mode, versus using a faster non-reasoning model and saving the cost and latency?
2. What's our process for re-evaluating model choice as new frontier releases ship, and how often do we actually revisit it?
3. For multimodal use cases under consideration, has accuracy been validated against our own documents/images, not just published benchmarks?

Technologies / Standards / Companies to Know: Anthropic Claude Opus/Sonnet/Haiku, OpenAI GPT-5.x, Google Gemini 3, model/system cards.

Recommended Learning:

- Introducing Claude Opus 4.5 (Anthropic): primary source release notes and capability framing.
- GPT-5.1 System Card Addendum (OpenAI): official capability and safety documentation.
- Gemini 3: Introducing the latest Gemini AI model (Google): official announcement.

Time Investment: 1 hour, ongoing (expect quarterly refresh as models update)

9.2 Open-Weight vs Proprietary Models

Priority: Must Understand

Executive Definition: Proprietary models (Claude, GPT, Gemini) are accessed only via API: you never hold the weights, and the provider controls updates, hosting, and terms. Open-weight models (Llama, DeepSeek, Mistral, Qwen) publish downloadable weights you can run on your own infrastructure, but "open" refers to distribution, not necessarily a permissive open-source license: terms vary and materially affect what you may do with them.

Why It Matters: Meta's Llama license is not OSI-approved open source: it carries usage conditions that require legal review, not a "free to use" assumption, according to the Open Source Initiative. Open-weight models trade frontier-tier convenience and vendor-managed reliability for control (on-prem/air-gapped deployment, no data leaving your infrastructure, no vendor roadmap dependency), but you take on hosting, scaling, and patching responsibility: this is a deployment-control and cost-structure decision, not simply a leaderboard comparison.

What I Need to Understand:

- "Open-weight" is not synonymous with "open source": read the actual license (Llama Community License, Mistral's various licenses, DeepSeek's license) before assuming unrestricted commercial use.
- Self-hosting an open-weight model means owning hosting, scaling, security patching, and update cadence: real infrastructure and staffing cost that a proprietary API abstracts away.
- Proprietary frontier models generally still lead on the hardest reasoning/multimodal tasks; open-weight models have closed much of the gap on everyday tasks at meaningfully lower inference cost.
- Air-gapped/on-prem or strict residency requirements are often the actual deciding factor for open-weight adoption, more than raw capability comparisons.

Questions I Should Be Able to Ask My Team:

1. For any open-weight model we deploy, has legal reviewed the license terms for our specific use case (commercial use, redistribution, data restrictions)?

2. What is the fully-loaded cost (infrastructure, staffing, patching) of self-hosting versus the equivalent proprietary API cost at our expected volume?
3. Which of our use cases have a genuine residency or air-gap requirement that only a self-hosted, open-weight model satisfies?

Technologies / Standards / Companies to Know: Meta Llama, DeepSeek, Mistral, Qwen, OpenAI gpt-oss, vLLM/TGI for self-hosted serving.

Recommended Learning:

- Llama 4 Community License Agreement: the actual license terms, not a summary.
- Meta's Llama license is still not Open Source (Open Source Initiative): a neutral, non-vendor analysis of why the licensing distinction matters.

Time Investment: 1 hour

9.3 Context Windows, Tokens & Inference Fundamentals

Priority: Should Understand

Executive Definition: Tokens are the units (roughly word-fragments) models process; a context window is the maximum number of tokens a model can consider at once, spanning the prompt, retrieved documents, conversation history, and the response together. Inference is running the model to produce output, with real cost and latency that scales with tokens processed: a larger context window doesn't make more tokens free, it just raises the ceiling.

Why It Matters: Context window size determines what fits in a single request, but a bigger window doesn't guarantee the model uses everything in it well: long, cluttered contexts measurably degrade output quality even within the stated limit, which is exactly why context engineering (final topic below) exists as a discipline. Cost and latency scale with tokens in both directions, so context window size is a real budget lever your team is choosing, not just a technical ceiling.

What I Need to Understand:

- Context window size sets an upper bound on what can be given to the model in one call: retrieval (RAG topic) exists precisely because you can't put an entire knowledge base in context.
- More context per request is not free: cost scales with tokens processed, and excessive or cluttered context can degrade accuracy even within the advertised limit.
- **Track, don't implement:** cost-per-request and latency-per-request trends, and whether accuracy holds as more of the context window is filled on your actual tasks.
- **Delegate to your team:** the exact tokenization scheme, embedding dimensionality, and low-level inference optimizations (batching, quantization, KV-caching): these are implementation detail, not executive-level decisions.
- Prompt caching (reusing a previously-processed prefix of tokens across repeated calls) is a real, delegable cost-optimization technique: you need to know it exists and ask whether it's used.

Questions I Should Be Able to Ask My Team:

1. What is our typical cost and latency per request today, and how does that change with the context sizes we actually use in production?
2. Are we using prompt caching or equivalent techniques for repeated large contexts (e.g., a shared system prompt or document set across many calls)?
3. Have we tested whether accuracy degrades as we fill more of the context window on our own tasks, rather than assuming the full advertised limit is usable at full quality?

Technologies / Standards / Companies to Know: Claude/GPT/Gemini context window tiers, prompt caching, tokenizers.

Recommended Learning:

- Understanding the context window (Anthropic docs): the primary technical reference.

Time Investment: 30 minutes

9.4 Model Routing, Fallback & Structured Output

Priority: Should Understand

Executive Definition: Model routing means automatically directing each request to the most appropriate model by cost, capability, or latency need, rather than hard-coding one model everywhere. Fallback means automatically retrying with an alternate model or region when the primary one is unavailable or degraded. Structured output means constraining a model's response to a defined schema (e.g., JSON) so downstream systems can reliably parse it without brittle text parsing.

Why It Matters: Committing an entire application to one model creates both a cost problem (using an expensive model for simple tasks) and a resilience problem (a single point of failure). Cloud providers now ship routing and failover as core infrastructure: AWS Bedrock's intelligent prompt routing and cross-region inference exist because production systems need this, not because it's a nice-to-have, and structured output has moved from "hope the model formats it right" to provider-enforced schema guarantees, a real reliability improvement over ad hoc text parsing.

What I Need to Understand:

- Routing by task (cheap/fast model for simple requests, frontier model for hard ones) is a cost-control lever your team should actively use, not a one-time setup.
- Fallback strategy (what happens when the primary provider is down or rate-limited) should be an explicit, tested design decision, not something discovered during an outage.
- Structured output (schema-constrained generation) is now natively supported by major providers and is materially more reliable than asking a model to "output JSON" in free text: insist on it wherever output is parsed programmatically.

- **Delegate to your team:** the specific routing algorithm, gateway product, and JSON-schema plumbing.
- **Track as an executive:** cost per model tier used, fallback incident frequency, and structured-output parse-failure rate.

Questions I Should Be Able to Ask My Team:

1. Do we route requests by task complexity/cost today, or does every request go to the same (likely most expensive) model regardless of need?
2. What actually happens when our primary provider has an outage or rate-limits us: is there a tested fallback, or does the application just fail?
3. For systems that parse model output automatically, are we using provider-enforced structured output, and what's our current parse-failure rate?

Technologies / Standards / Companies to Know: AWS Bedrock intelligent prompt routing / cross-region inference, OpenAI Structured Outputs, Anthropic tool use, LLM gateways (LiteLLM, Portkey).

Recommended Learning:

- Understanding intelligent prompt routing in Amazon Bedrock: official documentation on production routing/fallback.
- Structured Outputs (OpenAI API): the primary technical reference for schema-enforced output.

Time Investment: 1 hour

9.5 Fine-Tuning, Distillation, Small Language Models & Edge Inference

Priority: Monitor

Executive Definition: Fine-tuning further trains an existing model on your own examples so it performs a specific task better without repeating guidance in every prompt. Distillation trains a smaller "student" model to imitate a larger "teacher" model on a specific task, producing a cheaper, faster, narrower model. Small language models (SLMs) and edge inference mean running compact models locally (on a laptop, device, or private server) instead of calling a large model over the network.

Why It Matters: These techniques trade generality for cost, latency, and control: a fine-tuned or distilled small model can be dramatically cheaper for a narrow, stable, high-volume task than always calling a frontier model: OpenAI's own distillation tooling exists specifically to make "use the big model to teach a cheap model for one task" a supported production pattern, not a research exercise. This is the clearest "decide the strategy, delegate the mechanics" topic in this domain.

What I Need to Understand:

- Fine-tuning/distillation is justified for tasks that are narrow, high-volume, and stable: not for general-purpose or rapidly-changing tasks, where prompting or RAG against a general model is more practical.

- A fine-tuned model still needs the same data governance (see Data Classification topic) applied to its training data: it can memorize and leak sensitive examples.
- **Own as an executive:** which use cases justify the fixed cost of fine-tuning/distillation, what the ongoing retraining/maintenance commitment is, and whether the resulting model gets the same governance and access controls as any other production system.
- **Delegate to your team:** the actual training run, hyperparameters, and evaluation harness.
- A fine-tuned or distilled model is a new asset with its own lifecycle (versioning, retraining triggers, deprecation): it does not inherit automatic updates the way a hosted frontier model does.

Questions I Should Be Able to Ask My Team:

1. Which specific use cases have we identified as narrow and high-volume enough to justify fine-tuning or distillation instead of prompting a general model?
2. What's the retraining/maintenance commitment once we fine-tune or distill a model, and who owns it long-term?
3. Does the training data for any fine-tuned model meet the same classification and governance bar as our other data assets?

Technologies / Standards / Companies to Know: OpenAI supervised fine-tuning and distillation tooling, small Llama/Mistral/Qwen variants for edge, on-device runtimes (llama.cpp, ONNX Runtime).

Recommended Learning:

- Fine-tuning (OpenAI Developers guide): a primary, vendor-neutral-in-substance technical overview of when and how fine-tuning applies.
- Leveraging model distillation to fine-tune a model (OpenAI Cookbook): concrete illustration of the teacher/student pattern.

Time Investment: 1 hour

Prompt Engineering vs. Context Engineering

Last updated September 11, 2026 · 4 min read · https://erikcaldwell.com/field-guide/prompt-engineering-vs-context-engineering/

10.1 Prompt Engineering Fundamentals

Priority: Should Understand

Executive Definition: Prompt engineering is writing and structuring the instructions given directly to a model (clarity, examples, explicit format requests, role framing) to get more reliable output from a single request. It is a real, learnable skill with measurable effect on output quality, but it operates entirely within one call to the model and has no influence over what data, tools, or memory that call has access to.

Why It Matters: Prompt engineering still matters, and labs continuously publish updated, concrete guidance because specific technique reliably improves output quality. But it is a tactical, per-request skill: it cannot fix a request that lacks the right retrieved context, tool access, or permissions: that is the system-architecture problem covered next, which is where most enterprise AI quality gains now come from. Treat this as broad organizational literacy every knowledge worker should have, not a strategic investment area in itself.

What I Need to Understand:

- Well-documented, low-cost techniques (specific instructions, examples, explicit output format, letting the model reason before answering) measurably improve output quality and cost nothing to adopt broadly.
- Prompt engineering is bounded by what's in the single request: it cannot retrieve data the model wasn't given, cannot recall past sessions, and cannot call tools that weren't provided (that is Context Engineering's job: next topic).
- This is a skill for broad organizational literacy (every knowledge worker benefits from basic competence), not a specialized function to scale headcount around.
- Prompt-level gains are use-case-specific and often marginal at the system level compared to fixing retrieval, permissions, or tool access: don't over-invest here at the expense of context engineering.

Questions I Should Be Able to Ask My Team:

1. Do we have a basic, shared internal guide for effective prompting so common techniques are used consistently rather than reinvented per team?
2. When a use case underperforms, have we ruled out a system/context problem (retrieval, tools, permissions) before spending more effort on prompt wording?
3. Are we distinguishing ownership/versioning of fixed prompt templates embedded in applications from ad hoc end-user prompting in our governance approach?

Technologies / Standards / Companies to Know: Anthropic/OpenAI/Google prompting guides, few-shot prompting, chain-of-thought, prompt template/version management.

Recommended Learning:

- Prompt engineering best practices for 2026 (Anthropic): current, primary-source guidance.
- Claude prompting best practices (Claude Platform Docs): the technical reference version of the same guidance.

Time Investment: 30 minutes

10.2 Context Engineering & System Architecture Around the Model

Priority: Must Understand

Executive Definition: Context engineering is the discipline of deciding what information, tools, and memory a model actually receives for a given task (the retrieved documents, conversation history, tool definitions, and system instructions assembled around it) as distinct from prompt engineering, which only shapes the wording of the instruction itself. It treats the model as one component in a larger system whose overall design determines output quality far more than any single prompt does.

Why It Matters: This is the central point of this entire domain: as models have become broadly capable, the gap between good and poor enterprise AI outcomes has shifted from "which model" or "how the prompt is worded" to "what the system around the model actually gives it access to." Anthropic's own engineering guidance frames context engineering as the natural evolution of prompt engineering precisely because agentic and long-running tasks fail on context management (what's retrieved, what's remembered, what's pruned) not on phrasing. Every earlier topic in this domain (permissions-aware retrieval, data provenance, memory, model routing) is a context engineering concern; prompt engineering is one narrow, comparatively minor input to it.

What I Need to Understand:

- Context engineering covers everything that goes into a model's context at runtime (retrieved documents, tool outputs, conversation/memory, system instructions) curated and pruned deliberately rather than dumped in wholesale.
- Long-running agentic tasks fail primarily from context problems (irrelevant or excessive information crowding out what matters, stale memory, missing tool results), not from imperfect prompt wording: this is why "just improve the prompt" stops working as systems get more agentic.
- Good context engineering actively decides what to leave out as much as what to include; more context is not automatically better (see the degradation-with-length point in Context Windows).
- This is where investment should concentrate: retrieval quality, permission-aware data access, memory curation, and tool design will move enterprise AI outcomes far more than incremental prompt refinement.

- It requires cross-functional ownership (data, security, and application engineering together) not a single prompt author.

Questions I Should Be Able to Ask My Team:

1. When an AI application underperforms, do we diagnose it as a context problem (wrong/missing retrieval, stale memory, missing tool access) before treating it as a prompt or model problem?
2. Who owns end-to-end context assembly for our production AI systems: is it a defined engineering responsibility, or is it happening ad hoc inside prompts?
3. What share of our AI investment is going to retrieval/permissions/memory architecture versus prompt-level work, and does that match where the evidence says the returns actually are?

Technologies / Standards / Companies to Know: Anthropic context engineering guidance, agent frameworks (LangChain/LangGraph, LlamaIndex), Model Context Protocol (MCP), the retrieval and memory systems covered in the topics above.

Recommended Learning:

- Effective context engineering for AI agents (Anthropic): the primary source for this topic's framing and evidence.
- Contextual Retrieval (Anthropic): cross-reference: a concrete, foundational context-engineering technique.

Time Investment: 2-3 hours

Enterprise AI Gateway & Control Plane

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/enterprise-ai-gateway-and-control-plane/

11.1 AI Gateways & Model Gateways

Priority: Must Understand

Executive Definition: An AI gateway is a control-plane layer that sits between applications and the LLMs they call (hosted or third-party), giving the organization one place to enforce policy, meter usage, and standardize integration. Instead of every application team wiring its own SDK directly to a model provider, all traffic is proxied through a gateway that can authenticate callers, apply guardrails, log every request/response, and route to whichever underlying model backend is appropriate: functionally analogous to an API gateway in front of microservices, but for model calls instead of service calls.

Why It Matters: Without a gateway, every application team makes its own decisions about keys, logging, rate limits, and content safety: producing inconsistent security posture and no enterprise-wide visibility into what models are being called, by whom, and at what cost. A gateway is also the only practical place to retrofit policy and cost controls across dozens of applications without touching each one's code. Both AWS and Microsoft now ship this as first-party infrastructure (Bedrock's built-in security/observability layer, Azure's API Management (APIM) AI Gateway capabilities), signaling this has become a standard architectural layer, not an optional add-on.

What I Need to Understand:

- The gateway is a proxy/control plane, not a model: it does not change model quality, only how calls are authenticated, routed, observed, and constrained.
- Core gateway functions: auth/identity passthrough, request/response logging, rate limiting, retries/failover across providers, semantic caching, and unified API translation (so app code targets one interface regardless of underlying model).
- Cloud-native versions (Azure API Management's GenAI gateway capabilities, Bedrock) integrate natively with existing Identity and Access Management (IAM) and network controls; independent gateways (LiteLLM, Kong AI Gateway, Portkey) trade that native integration for multi-cloud/multi-provider flexibility.
- A gateway is the natural enforcement point for centralized policy controls and the telemetry source for observability work: it should not be evaluated in isolation from those.
- Gateway choice interacts directly with build-vs-buy and vendor lock-in decisions (see Multi-Model Strategy topic).

Questions I Should Be Able to Ask My Team:

1. Is every production LLM call in the organization routed through a single gateway, or can teams still call provider APIs directly and bypass it?

2. What happens to in-flight requests and cost/policy enforcement if the gateway itself has an outage: is there a documented failure mode?
3. Does the gateway log full prompts/responses by default, and who can access those logs?

Technologies / Standards / Companies to Know: AWS Bedrock (native gateway + guardrails), Azure AI Foundry / Azure API Management GenAI gateway capabilities, Google Cloud API Gateway model routing / Apigee, LiteLLM, Kong AI Gateway, Portkey.

Recommended Learning:

- Access Foundry Models and Other Language Models Through a Gateway: Azure Architecture Center: Microsoft's own reference architecture for why and how to front model calls with a gateway.
- AI gateway capabilities in Azure API Management: official documentation of what a gateway layer is expected to enforce (token limiting, caching, load balancing).
- Security, Guardrails, and Observability in Amazon Bedrock: AWS's architecture documentation for its built-in gateway/control functions.
- Overview of model routing: Google Cloud API Gateway: Google's documentation on routing gateway traffic across model backends.

Time Investment: 1 hour

11.2 Centralized Policy Enforcement (authZ, rate limits, Data Loss Prevention (DLP), content filtering, cost controls)

Priority: Must Understand

Executive Definition: Centralized policy enforcement means authorization, rate limiting, data-loss-prevention (DLP), content filtering, and cost controls are applied consistently at the gateway layer rather than left to individual application teams to implement (or forget to implement). This covers who is allowed to call which model with what data (authZ), how much traffic and spend any caller can generate (rate/cost limits), what sensitive data is blocked from leaving or entering the model (DLP), and what categories of content the model is disallowed from producing or acting on (content filtering).

Why It Matters: Decentralized enforcement produces the same outcome every time: some team's integration has weaker controls than the others, and that becomes the incident. OWASP's Top 10 for LLM Applications names excessive agency, sensitive information disclosure, and improper output handling among the leading LLM application risks specifically because these are usually per-application gaps rather than platform-wide policy failures. Centralizing enforcement turns a governance requirement into a configuration change in one place instead of a code change in fifty places.

What I Need to Understand:

- AuthZ at the gateway should be scoped beyond "can this caller reach the model" to "can this caller's identity/role access this specific model, dataset, or tool": this is the same excessive-agency risk OWASP flags for agentic systems.
- DLP for LLM traffic means inspecting both inbound prompts (for secrets, customer Personally Identifiable Information (PII) being pasted into prompts) and outbound completions (for PII/regulated data being generated): this is materially different from traditional network DLP, which does not understand model context.
- Content filtering (both AWS Bedrock Guardrails and Vertex AI's safety filters implement this as configurable severity thresholds per harm category, not a single on/off switch) needs to be tuned per use case: a legal-research assistant and a customer support bot need different filter thresholds.
- Rate limits and cost controls at this layer are the first line of defense against runaway-cost scenarios, and should be enforced per identity/application, not just globally.
- Policy enforcement needs its own audit trail: decisions to block, redact, or allow need to be logged for incident-response and audit-replay purposes.

Questions I Should Be Able to Ask My Team:

1. Are our DLP rules inspecting both prompts and completions, or only one direction?
2. Which OWASP LLM Top 10 categories does our current guardrail configuration actually cover, and which are we accepting as unmitigated risk?
3. Can any single application or identity exceed its rate/cost limit by routing around the gateway, and how would we detect that?

Technologies / Standards / Companies to Know: OWASP Top 10 for LLM Applications, NIST AI RMF (AI 600-1 Generative AI Profile), Amazon Bedrock Guardrails, Azure AI Content Safety, Google Model Armor / Vertex AI safety filters.

Recommended Learning:

- OWASP Top 10 for LLM Applications 2025: the canonical risk taxonomy this control layer is meant to mitigate.
- Block denied topics to help remove harmful content: Amazon Bedrock and Remove PII from conversations by using sensitive information filters: concrete documentation of DLP/content-filter mechanics.
- Safety and content filters: Vertex AI: how threshold-based content filtering is actually configured.
- NIST AI 600-1: Artificial Intelligence Risk Management Framework: Generative AI Profile: the governance framework these controls are typically mapped to for audit purposes.

Time Investment: 2-3 hours

11.3 Gateway Architecture Choices (AWS Bedrock, Azure AI, Google Vertex AI, direct model APIs, build vs buy)

Priority: Monitor

Executive Definition: Enterprises choosing a gateway approach are really choosing among four architecture patterns: a cloud provider's native gateway tied to that provider's models and IAM (AWS Bedrock, Azure AI Foundry), a cloud-native but multi-model gateway (Vertex AI plus Google Cloud API Gateway's model routing, or Azure API Management sitting in front of both Azure and non-Azure models), an independent multi-cloud gateway (LiteLLM, Kong, Portkey) that is provider-agnostic but adds an operational component you now own, or direct model API calls with no gateway at all. Each trades integration depth against lock-in and flexibility.

Why It Matters: This decision determines how hard it is to add a second model provider later, how much of your existing cloud IAM/networking investment you can reuse, and who is on the hook when the gateway itself has an incident. It is an architecture decision with multi-year switching costs, not a procurement checkbox: treating it as the latter is how organizations end up locked into a single model provider's roadmap for reasons that were never actually evaluated.

What I Need to Understand:

- Native cloud gateways (Bedrock, Azure AI Foundry's model router and APIM AI Gateway) give you tighter IAM/VPC/logging integration but couple your model strategy to that cloud's model catalog and release cadence.
- Vertex AI plus Google Cloud API Gateway follows the same native pattern on GCP, including model routing across Gemini and third-party models hosted there.
- Independent gateways decouple model choice from cloud provider but add a new piece of infrastructure your team must operate, secure, and scale: the gateway becomes a new single point of failure and a new attack surface.
- "Direct API" (no gateway) is viable only at very small scale or for a single, tightly scoped application; it does not survive contact with more than a handful of applications or any centralized policy requirement.
- Build-vs-buy here is really "adopt the cloud-native layer we're already paying for" vs. "add an independent control plane for multi-cloud flexibility": there is rarely a case for building this from scratch given how much this space has matured.

Questions I Should Be Able to Ask My Team:

1. If we standardize on one cloud provider's native gateway, what is our actual plan and cost to add a second model provider later?
2. What is the operational owner and SLA for our gateway layer: is it treated with the same rigor as any other tier-1 production dependency?
3. Does our gateway choice let security apply the same authZ/DLP policy regardless of which underlying model is called, or does policy have to be re-implemented per provider?

Technologies / Standards / Companies to Know: AWS Bedrock, Azure AI Foundry / Azure API Management, Google Vertex AI / Google Cloud API Gateway, LiteLLM, Kong AI Gateway, Portkey.

Recommended Learning:

- Access Foundry Models and Other Language Models Through a Gateway: Azure Architecture Center: Microsoft's explicit reasoning for a gateway tier versus direct calls.
- Model router for Microsoft Foundry: concepts: how native multi-model routing is architected on one cloud platform.
- Overview of model routing: Google Cloud API Gateway: the equivalent native pattern on GCP.
- Security, Guardrails, and Observability in Amazon Bedrock: the AWS-native reference point for comparison.

Time Investment: 2-3 hours

AI Evaluation & Quality Engineering

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/ai-evaluation-and-quality-engineering/

12.1 AI Evals & Golden Datasets

Priority: Must Understand

Executive Definition: An "eval" is a repeatable test that scores model or agent output against a defined standard (accuracy, relevance, tone, safety) usually run automatically against a curated "golden dataset" of representative inputs with known-good (or known-acceptable) outputs. Evals are the AI-era equivalent of a test suite, but because LLM outputs are non-deterministic and rarely have one single correct answer, evals typically score against a rubric or a reference answer within a tolerance, rather than an exact match.

Why It Matters: Conventional software testing assumes a deterministic function: same input, same output, pass/fail. LLM outputs vary between runs even at identical settings, and "correct" is frequently a matter of degree (a legally accurate but poorly worded answer vs. a fluent but subtly wrong one). Without golden datasets and evals, model or prompt changes ship on faith: the only signal that something regressed is a user complaint, by which point it is already in production. This is the foundational competency underneath every other evaluation topic below.

What I Need to Understand:

- A golden dataset is a curated, versioned set of representative inputs (ideally including edge cases and known failure modes) with reference outputs or acceptance criteria: it needs the same change control and ownership as a codebase, not a one-time spreadsheet.
- Evals fall into three broad families: exact/programmatic checks (does the output contain a required field, pass a regex, parse as valid JSON), similarity-based checks (embedding or n-gram overlap against a reference answer), and LLM-as-judge checks (see below): each has different cost, reliability, and failure characteristics.
- Because there is often no single correct answer, most production evals score against a rubric with partial credit rather than binary pass/fail, and rubric design is itself a skill that requires domain expertise, not just engineering effort.
- Golden datasets go stale as the product and real user traffic evolve: production evaluation (see continuous evaluation topic below) exists precisely because a static golden dataset cannot cover what real users will actually do.
- Eval quality is only as good as the dataset's coverage of realistic failure modes; a dataset built only from happy-path examples will pass models that fail badly on messy real input.

Questions I Should Be Able to Ask My Team:

1. Who owns and updates our golden datasets, and how often are they refreshed against real production traffic and known failure cases?
2. What percentage of our golden dataset examples were sourced from actual production incidents versus hypothetical test cases?
3. When we change a prompt, model, or RAG pipeline, is there a required eval run and pass threshold before it ships, or is this discretionary?

Technologies / Standards / Companies to Know: Retrieval Augmented Generation Assessment (RAGAS), DeepEval, Langfuse, Databricks MLflow evaluation, promptfoo.

Recommended Learning:

- RAGAS: Automated Evaluation of Retrieval Augmented Generation (arXiv): the original academic paper behind one of the most widely adopted open-source eval frameworks.
- LLM Evaluation: Best Practices and Methods: Databricks Engineering Blog: a vendor-neutral, engineering-depth treatment of dataset construction and metric selection.
- Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena (arXiv): foundational paper establishing why rubric/judge-based evaluation replaces exact-match testing for open-ended output.

Time Investment: Half day

12.2 Trajectory Evaluation for Agents & Tool-Call Correctness

Priority: Should Understand

Executive Definition: Trajectory evaluation scores not just an agent's final answer but the full sequence of decisions it took to get there: which tools it called, in what order, with what arguments, and whether each step was necessary and correct. A single-turn chat eval only has to check the final output; an agent eval has to check the path, because two agents can reach the same correct final answer while one took an efficient, safe route and the other took an expensive, unauthorized, or unsafe one.

Why It Matters: An agent that books the wrong flight but then "corrects" it might show a correct final trajectory-blind eval score while having generated duplicate charges, hit an external API twice, or briefly held incorrect state that a downstream system already acted on. Final-answer-only evaluation is blind to exactly the failure modes (wasted tool calls, wrong tool selection, unauthorized actions taken and then walked back) that drive cost overruns and security incidents in agentic systems.

What I Need to Understand:

- Trajectory correctness typically evaluates three separate things: tool selection (did it pick the right tool), argument correctness (did it call the tool with valid, correct parameters), and step efficiency/necessity (did it take an unnecessarily long or costly path, including retries and backtracking).

- Academic benchmarks such as TRAJECT-Bench (arXiv:2510.04550) formalize trajectory-aware scoring specifically because prior agent benchmarks judged only the end state and missed these path-level failures.
- Multi-step agent trajectories compound error: a small tool-argument mistake early in a chain can cascade into a completely wrong final answer that nonetheless "looks" plausible, which is why trajectory-level evals catch failures that final-answer evals miss.
- Trajectory evaluation depends on having full step-by-step tracing (see AgentOps topic below): you cannot score a path you did not capture.
- This is a newer and less standardized discipline than single-turn LLM evaluation; expect your team's tooling and rubrics here to be more custom-built than off-the-shelf.

Questions I Should Be Able to Ask My Team:

1. Do our agent evals score the full tool-call trajectory, or only whether the final answer was correct?
2. What is our defined "acceptable" number of tool calls or retries per task, and do we flag trajectories that exceed it even when the final answer is right?
3. Can we replay a failed agent trajectory end-to-end from stored traces, and how far back does that trace history go?

Technologies / Standards / Companies to Know: TRAJECT-Bench, AgentOps, LangSmith, Arize.

Recommended Learning:

- TRAJECT-Bench: A Trajectory-Aware Benchmark for Evaluating Agentic Tool Use (arXiv): the most directly relevant academic framing of trajectory-level agent evaluation.
- AgentOps: Enabling Observability of LLM Agents (arXiv): academic treatment of what needs to be captured to make trajectories evaluable at all.
- LLM Agent Evaluation Metrics: Tool Calling, Task Completion, Reasoning, and Trace-Based Evals: Confident AI: practitioner-level breakdown of the metric families used in production trajectory evals.

Time Investment: 2-3 hours

12.3 Hallucination, Groundedness & LLM-as-Judge

Priority: Must Understand

Executive Definition: Hallucination is a model generating confident, fluent output that is factually wrong or unsupported by its source material. Groundedness is the inverse property being measured for: whether an output can be traced back to and is fully supported by the retrieved context or source documents it was given (most relevant in retrieval-augmented generation). LLM-as-judge is the now-standard practice of using a separate (often stronger) model to score another model's output against a rubric, because human review does not scale to production volumes and exact-match testing does not work on open-ended text.

Why It Matters: Hallucination is not a bug that gets patched away; it is a structural property of how these models generate text, and it varies significantly by task. Public leaderboards such as Vectara's hallucination leaderboard track measured hallucination rates across models specifically because this remains an active, unsolved, model-dependent risk, not a solved problem you can assume away. LLM-as-judge is powerful but introduces its own failure mode: judge models have documented biases (verbosity bias, position bias, self-preference) that were characterized in the original MT-Bench paper (arXiv:2306.05685) and must be accounted for, not ignored.

What I Need to Understand:

- Groundedness/faithfulness scoring (as formalized in frameworks like RAGAS) checks whether every claim in an output is supported by the retrieved context: this is the primary defense against hallucination in RAG systems specifically, and does not apply to open-domain generation without retrieval.
- LLM-as-judge is a measurement tool, not ground truth: judge models need their own validation against human-labeled samples periodically, or scoring drift goes undetected.
- Known LLM-as-judge biases include favoring longer answers, favoring the first answer shown in pairwise comparison, and favoring outputs stylistically similar to the judge's own outputs: mitigations include randomizing order, using multiple judges, and reference-based (not just pairwise) scoring.
- Hallucination rate is not a single number for a model; it varies by task type and is actively tracked and updated.
- Groundedness/hallucination checks should run both in pre-production evals and continuously in production (see continuous evaluation topic below), since retrieval quality and real user queries shift over time in ways a static eval set won't catch.

Questions I Should Be Able to Ask My Team:

1. What groundedness or faithfulness metric are we using in production, and what threshold triggers a flagged response?
2. If we use an LLM judge, how was it validated against human judgment, and how often is that validation refreshed?
3. Do we distinguish between hallucination in retrieval-grounded tasks (unsupported claims) versus open-domain tasks (factually wrong claims), since the mitigation for each is different?

Technologies / Standards / Companies to Know: RAGAS, Vectara's Hughes Hallucination Evaluation Model (HHEM), TruthfulQA, DeepEval, G-Eval-style judge prompting.

Recommended Learning:

- Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena (arXiv): the foundational paper documenting judge-model biases and validating judge agreement against human raters.
- RAGAS: Automated Evaluation of Retrieval Augmented Generation (arXiv): defines faithfulness/groundedness scoring methodology used across most modern RAG eval tooling.

- HHEM v2: A New and Improved Factual Consistency Scoring Model: Vectara: a purpose-built open hallucination-detection model, with methodology detail beyond a marketing claim.
- vectara/hallucination-leaderboard (GitHub): continuously updated, methodology-documented cross-model hallucination comparison.

Time Investment: Half day

12.4 Red Teaming, Adversarial & Continuous Production Evaluation

Priority: Must Understand

Executive Definition: Red teaming is deliberately attacking your own AI system (prompt injection, jailbreaks, data exfiltration attempts) before an adversary does. Continuous production evaluation is the parallel discipline of testing changes safely once the system is live, using techniques like shadow testing (running a new model/prompt on real traffic without serving its output to users), canary deployments (serving the new version to a small percentage of real users first), and A/B testing (serving two versions to different user segments and comparing outcomes). Together these replace the false assumption that passing a pre-launch eval means a system is safe indefinitely.

Why It Matters: LLM systems fail in ways static testing does not anticipate: adversarial prompt injection tops the OWASP LLM Top 10, and model behavior drifts as providers update models behind the scenes, as retrieval corpora change, and as real user query patterns diverge from the golden dataset. Shadow and canary deployment patterns exist because rolling out a new model or prompt version directly to 100% of production traffic, with no fallback, is the same category of risk that canary deployment already solved for traditional software: except the failure modes here (a subtly worse groundedness rate, a new jailbreak surface) are much harder to catch with traditional health checks alone.

What I Need to Understand:

- Red teaming for LLM/agent systems should be continuous, not a pre-launch audit: OWASP's GenAI Red Teaming initiative and NIST's AI RMF both frame adversarial testing as an ongoing control, not a gate you pass once.
- Shadow testing captures real production output for comparison without any user-facing risk: it is the lowest-risk way to validate a new model/prompt version against real traffic patterns your golden dataset does not cover.
- Canary deployment (serving to a small real-user percentage) tests actual user impact but carries real risk to that percentage of users, so it needs an automatic rollback trigger tied to the same eval metrics as pre-production testing.
- A/B testing for LLM changes needs outcome metrics beyond user satisfaction: cost per outcome, hallucination/groundedness rate, and trajectory correctness should all be tracked per arm, not just top-line engagement.

- Adversarial testing scope should explicitly include agentic risks (an agent being manipulated into taking unauthorized actions via injected instructions in retrieved content or tool outputs), which is a distinct and newer attack surface from classic prompt jailbreaks.

Questions I Should Be Able to Ask My Team:

1. Do we red-team continuously in production, or only once before initial launch?
2. What is our rollback trigger and threshold for a canary deployment that starts showing degraded groundedness or elevated hallucination scores?
3. For agentic systems specifically, have we tested prompt injection delivered through retrieved documents or tool outputs, not just through direct user input?

Technologies / Standards / Companies to Know: OWASP GenAI Red Teaming Initiative, NIST AI RMF (AI 600-1), promptfoo, Microsoft PyRIT.

Recommended Learning:

- AI Red Teaming Initiative: OWASP Gen AI Security Project: the primary open framework for structuring adversarial testing programs.
- NIST AI 600-1: Generative AI Profile: governance-level framing of continuous risk testing as a required control, not a one-time exercise.
- Shadow deployment vs. canary release of machine learning models: a practitioner-level engineering explanation of the mechanics and tradeoffs between the two patterns.

Time Investment: Half day

AI Security

Last updated September 11, 2026 · 17 min read · https://erikcaldwell.com/field-guide/ai-security/

13.1 OWASP Top 10 for LLM Applications

Priority: Must Understand

Executive Definition: The OWASP Top 10 for LLM Applications is the closest thing the industry has to a standard taxonomy of security failure modes in systems built on large language models: analogous to OWASP's long-standing web application Top 10. The current version (LLM01–LLM10:2025) covers risks from prompt injection and data disclosure through supply chain, output handling, and excessive agency. It gives your organization shared vocabulary for risk registers, vendor questionnaires, and audit findings.

Why It Matters: Without a common taxonomy, every team, vendor, and auditor invents their own language for the same failure modes, which makes risk comparable across projects nearly impossible and lets real gaps hide behind marketing terms like "AI-safe" or "enterprise-grade." This list is also the reference frame the rest of this curriculum's security topics map onto: treat it as the index, not a standalone checklist (OWASP GenAI Security Project, "OWASP Top 10 for LLM Applications 2025").

What I Need to Understand:

- The ten categories: LLM01 Prompt Injection, LLM02 Sensitive Information Disclosure, LLM03 Supply Chain, LLM04 Data and Model Poisoning, LLM05 Improper Output Handling, LLM06 Excessive Agency, LLM07 System Prompt Leakage, LLM08 Vector and Embedding Weaknesses, LLM09 Misinformation, LLM10 Unbounded Consumption.
- This list describes application-layer risk in LLM-based systems, not general model safety/alignment research: it is written for builders and buyers of LLM applications.
- OWASP's GenAI Security Project has since published follow-on, agent-specific guidance (Agentic AI Threats and Mitigations, Securing Agentic Applications Guide) because the original Top 10 predates widespread tool-calling agents: the two should be read together.
- The list changes: version 2025 reordered and renamed several 2023 categories (e.g., "Insecure Output Handling" became "Improper Output Handling"); expect further revision as agentic deployments mature.
- It is a risk taxonomy, not a certification: passing an "OWASP LLM Top 10 scan" from a vendor tool does not mean an application is secure.

Questions I Should Be Able to Ask My Team:

1. Which of the ten LLM categories have we explicitly assessed for our production AI systems, and which have we not looked at yet?
2. Do our vendor security questionnaires and RFPs reference this taxonomy, or are we accepting vague "we take AI security seriously" answers?

3. Are we tracking the newer OWASP agentic-AI guidance separately, since most of our real exposure now comes from agents that act, not from chatbots that only generate text?

Technologies / Standards / Companies to Know: OWASP GenAI Security Project (formerly OWASP Top 10 for LLM Applications); OWASP AI Exchange (owaspai.org); NIST AI RMF / NIST AI 600-1 Generative AI Profile as the complementary US government framework.

Recommended Learning:

- OWASP Top 10 for LLM Applications 2025: the primary source document itself.
- OWASP Gen AI Security Project: Resources: index of the full family of guides, including the agentic-specific ones.
- NIST AI Risk Management Framework: the US government's parallel, broader risk framework.

Time Investment: 1 hour

13.2 Prompt Injection (Direct & Indirect)

Priority: Must Understand

Executive Definition: Prompt injection is the act of getting an LLM to follow attacker-supplied instructions instead of (or in addition to) its developer's instructions. "Direct" injection means the attacker types the malicious instruction straight into the chat. "Indirect" injection is more dangerous: the malicious instruction is hidden inside content the AI reads as data (a web page, an email, a file, a support ticket, a calendar invite) and the model can't reliably distinguish "instructions from my operator" from "text I was asked to summarize." No fully reliable technical fix exists as of 2026 (OWASP LLM01:2025; Simon Willison, "The lethal trifecta for AI agents," 2025).

Why It Matters: This is the foundational vulnerability of the entire LLM security field: nearly every other item on this list (excessive agency, memory poisoning, insecure output handling, MCP supply-chain risk) is either a variant of prompt injection or a mechanism that makes its consequences worse. It is also unsolved: model providers have reduced susceptibility but have not eliminated it, so your controls must assume it will sometimes succeed, not that it can be prevented.

What I Need to Understand:

- The distinction between direct injection (adversarial user) and indirect injection (adversarial content the AI processes on someone else's behalf): indirect injection is the higher-risk case for enterprise deployments because the "attacker" never talks to your system at all.
- Simon Willison's "lethal trifecta": an agent is dangerous when it simultaneously has (1) access to private/sensitive data, (2) exposure to untrusted content, and (3) a way to communicate externally (send email, post to the web, call an API). Removing any one leg breaks the attack.
- Prompt injection cannot currently be reliably filtered out the way SQL injection can be parameterized away: defenses are architectural (limiting what the agent can do after reading untrusted content), not purely input-sanitization.

- This is why "excessive agency" is the risk that turns a prompt injection from an annoyance into a breach: injection is the entry point, agency is what makes it costly.
- Real disclosed incidents exist, not just theory (see MCP Supply-Chain Risk topic): this is operational risk today, not speculative research.

Questions I Should Be Able to Ask My Team:

1. For each agent we operate, can you draw its "lethal trifecta" (what private data it can see, what untrusted content it processes, and what external actions it can take) and what breaks that chain?
2. When an agent reads content from an external or user-controlled source (a webpage, an email, an uploaded document), what stops embedded instructions in that content from being treated as commands?
3. What is our incident response plan the day a customer or researcher reports a working prompt injection against one of our AI-enabled products?

Technologies / Standards / Companies to Know: OWASP LLM01:2025; Simon Willison (independent researcher who named and has extensively documented prompt injection since 2022); Google DeepMind and Microsoft published layered-defense research for Gemini/Copilot; NIST AI 600-1.

Recommended Learning:

- The lethal trifecta for AI agents: Simon Willison's framing, the clearest executive-level mental model available.
- OWASP LLM01:2025 Prompt Injection: the formal taxonomy entry with attack examples.
- Prompt injection: Wikipedia: solid neutral overview and history for a first read.
- Simon Willison's prompt-injection archive: running log of real disclosed cases since 2022, useful for staying current.

Time Investment: 2-3 hours

13.3 Excessive Agency & Agent Goal Hijacking

Priority: Must Understand

Executive Definition: Excessive Agency is what happens when an LLM-based system is granted more autonomy, tool access, or permission than it needs, so that when it is manipulated (via prompt injection) or simply makes a reasoning error, it can take a damaging real-world action rather than just produce a wrong sentence. "Goal hijacking" is the specific case where an attacker redirects an agent's actual objective mid-task, using injected instructions the agent treats as legitimate updates to its goal (OWASP LLM06:2025).

Why It Matters: This is the topic that separates chatbot-era risk from agent-era risk: a hallucinated sentence in a chat window is embarrassing, but a hallucinated (or hijacked) tool call that sends a wire transfer, deletes records, or emails a customer is a security and financial incident. Every mitigation here

assumes prompt injection will sometimes succeed: the question is what the agent is *able* to do once it does.

What I Need to Understand:

- OWASP frames excessive agency as a function of three things you control directly: excessive functionality (tools the agent has but doesn't need for its task), excessive permissions (each tool grants more access than the task requires), and excessive autonomy (actions taken without human checkpoints).
- Goal hijacking is not the same failure as a bad prompt: it happens *during* execution, often via content the agent reads mid-task (a tool result, a document, a webpage), which is why it is closely tied to indirect prompt injection.
- Multi-step, multi-tool agents compound this risk: each additional tool and each additional autonomous step is another opportunity for a hijacked goal to translate into an action.
- The practical fix is architectural, not behavioral: scope each agent's tools and permissions to the minimum needed per task, and require human or policy checkpoints before consequential actions: you cannot prompt your way out of this.
- OWASP's Agentic AI Threats and Mitigations guide extends this analysis specifically for multi-agent and tool-calling systems, beyond the original single-model Top 10 framing.

Questions I Should Be Able to Ask My Team:

1. For our production agents, what is the largest single action (financial, data-modifying, communication-sending) each one is technically capable of taking without a human in the loop?
2. Have we mapped each agent's tool list against the tasks it actually performs, or did tools accumulate because they were "useful to have"?
3. What checkpoint (approval, transaction limit, anomaly detection) sits between "agent decides to act" and "action executes" for our highest-risk agents?

Technologies / Standards / Companies to Know: OWASP LLM06:2025 Excessive Agency; OWASP "Agentic AI: Threats and Mitigations"; OWASP "Securing Agentic Applications Guide."

Recommended Learning:

- OWASP LLM06:2025 Excessive Agency: formal definition and the functionality/permissions/autonomy framing.
- Agentic AI: Threats and Mitigations: OWASP's agent-specific extension of this risk.
- Securing Agentic Applications Guide 1.0: practical mitigations for multi-tool, multi-step agents.

Time Investment: 1 hour

13.4 Memory, Context & RAG Poisoning

Priority: Should Understand

Executive Definition: Memory and RAG (retrieval-augmented generation) poisoning is the injection of malicious or false content into the data an AI system retrieves or remembers (a vector database, a knowledge base, a long-term memory store, or conversation history) so that the poisoned content later shapes the model's outputs or an agent's decisions. Unlike prompt injection, which manipulates a single interaction, poisoning corrupts the system's persistent knowledge, so the effect can be delayed, repeated, and harder to trace back to a root cause (OWASP LLM04:2025 Data and Model Poisoning; OWASP LLM08:2025 Vector and Embedding Weaknesses).

Why It Matters: As enterprises give agents persistent memory and connect them to internal knowledge bases for accuracy, they create a new class of asset (the memory/RAG store) that is rarely protected with the same rigor as production databases, even though corrupting it can silently redirect agent behavior across many future sessions, not just one. Academic red-teaming has already demonstrated this against real agent architectures (Chen et al., "AgentPoison," NeurIPS 2024).

What I Need to Understand:

- Poisoning can enter through content your organization doesn't fully control: documents indexed for RAG, tickets or emails an agent is told to "remember," or shared knowledge bases with broad write access.
- This overlaps with but is distinct from prompt injection: injection manipulates the live conversation; poisoning manipulates the durable store the model consults later, so a single successful attack can affect every subsequent user or session that touches that data.
- AgentPoison and similar research show that a small number of crafted "trigger" documents can reliably backdoor an agent's retrieval behavior while remaining hard to detect through normal content review.
- Mitigations resemble data-supply-chain controls more than classic AI safety controls: provenance tracking, write-access restriction on memory/vector stores, periodic integrity auditing, and treating anything an agent "learns" from unvetted sources as untrusted until reviewed.
- Long-lived agent memory (an agent that updates its own notes/preferences over time) is the newest and least-audited version of this risk: most organizations have no process yet for reviewing what an agent has "decided to remember."

Questions I Should Be Able to Ask My Team:

1. Who can write to the vector databases and memory stores our agents read from, and is that access controlled as tightly as write access to production data?
2. Do we have any process for auditing what an agent has stored in long-term memory, or is it an opaque, self-updating black box?
3. If we discovered a poisoned document in our RAG index today, could we identify which agent outputs or actions it had already influenced?

Technologies / Standards / Companies to Know: OWASP LLM04:2025 (Data and Model Poisoning), LLM08:2025 (Vector and Embedding Weaknesses); AgentPoison (academic red-teaming framework, NeurIPS 2024); vector database access-control features (increasingly offered by Pinecone, Weaviate,

Milvus, and cloud-native vector stores).

Recommended Learning:

- AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases: the primary academic source demonstrating this attack class against real agent memory/RAG systems.
- OWASP LLM04:2025 Data and Model Poisoning: formal taxonomy entry.
- OWASP LLM08:2025 Vector and Embedding Weaknesses: the RAG-specific companion category.

Time Investment: 1 hour

13.5 Insecure Output Handling & Unexpected Code Execution

Priority: Should Understand

Executive Definition: Improper (formerly "insecure") output handling is the failure to validate, sanitize, or scope-check an LLM's output before it is passed to a downstream system: a browser, a shell, a database query, another application. Because an LLM's output is attacker-influenceable (via prompt injection), treating that output as trusted input to an execution context can let an attacker achieve classic outcomes (code execution, SQL injection, cross-site scripting) through the model as the delivery mechanism, even when the underlying downstream system has no LLM-specific vulnerability at all (OWASP LLM05:2025 Improper Output Handling).

Why It Matters: Many teams building on LLMs correctly harden their input pipeline against prompt injection but then pipe the model's raw output straight into an interpreter, a code execution sandbox, or a database call, recreating decades-old injection vulnerabilities with a new front door. This is a direct engineering discipline gap, not a novel AI risk: the fix is the same output-validation rigor already required for any untrusted-input pipeline.

What I Need to Understand:

- The core principle: model output must be treated as untrusted input to whatever system consumes it next: the same standard applied to user input in traditional web security.
- This is especially acute for code-generation and code-execution agents (e.g., coding assistants that run the code they write), where the model's output is literally executable.
- The risk compounds with excessive agency: an agent that both generates and executes its own output, with broad tool access, removes the human review step that used to catch bad output before it ran.
- Standard mitigations are conventional secure-coding practices adapted to this context: output encoding appropriate to the destination (HTML, SQL, shell), sandboxing code execution, least-privilege execution contexts, and never using string-concatenation of model output into privileged commands.
- This category was renamed from "Insecure Output Handling" (2023) to "Improper Output Handling" (2025) with a broadened scope: a signal of how fast this area is still being refined.

Questions I Should Be Able to Ask My Team:

1. Anywhere our systems execute code, run queries, or render content generated by an LLM, what validation happens between the model's output and that execution?
2. For our coding agents specifically, what sandbox isolates code the model writes and runs, and what can that sandbox reach (network, file system, credentials)?
3. Have we audited our LLM-output pipelines the same way we'd audit any pipeline that takes untrusted user input, or did AI get a pass because "it's just the model"?

Technologies / Standards / Companies to Know: OWASP LLM05:2025 Improper Output Handling; standard secure-coding practices (OWASP ASVS) applied to model output; sandboxing/isolation approaches used by coding-agent vendors.

Recommended Learning:

- OWASP LLM05:2025 Improper Output Handling: formal definition.
- OWASP Top 10 for LLM Applications 2025: full document for cross-referencing this category against excessive agency and prompt injection.

Time Investment: 1 hour

13.6 MCP & Tool/Plugin Supply-Chain Risk

Priority: Must Understand

Executive Definition: The Model Context Protocol (MCP) and similar tool/plugin frameworks let AI agents call external tools and data sources through a standardized interface: which also means a malicious, compromised, or merely careless MCP server can feed an agent poisoned instructions, exfiltrate data the agent has access to, or execute arbitrary commands on the host running it. Because agents typically trust what their configured tools tell them, a compromised tool is a compromised agent, and this risk has moved from theoretical to repeatedly, publicly disclosed within about eighteen months of MCP's release (National Security Agency (NSA)/Cybersecurity and Infrastructure Security Agency (CISA), "Model Context Protocol (MCP) Security," 2026).

Why It Matters: MCP servers are effectively a new, fast-growing software supply chain: often installed with a one-line command from a public registry, frequently maintained by small teams or individuals, and granted access to source code, email, financial systems, or credentials by design. Multiple real, disclosed incidents (not hypothetical research) already exist: a May 2025 flaw in the official GitHub MCP server let a malicious GitHub issue trigger exfiltration of private repository data via prompt injection (Invariant Labs); a command-injection vulnerability in the widely used `mcp-remote` package, logged as Common Vulnerabilities and Exposures (CVE) entry CVE-2025-6514, affected hundreds of thousands of downloads; and a cloned, trojanized MCP package was found distributing information-stealing malware via a public registry in early 2026.

What I Need to Understand:

- MCP servers should be evaluated with the same supply-chain rigor as any third-party dependency: provenance, maintainer trustworthiness, update cadence, and what permissions/data access they request: "it's just a small open-source connector" is exactly the profile of past supply-chain compromises in traditional software.
- The GitHub MCP incident illustrates a general pattern (a "toxic agent flow," per Invariant Labs): an agent with access to both a public, attacker-writable surface (issues, tickets, comments) and a private, sensitive data source is exploitable regardless of how well the underlying tool is coded: this is the lethal trifecta instantiated through a tool integration.
- STDIO-transport MCP servers run as local processes with the same privileges as the user or service running them: a compromised or malicious one is not sandboxed by default unless you explicitly isolate it.
- Public MCP server registries (analogous to npm or PyPI) currently have limited vetting; typosquatting and cloned-malicious packages are an active, documented pattern, not a hypothetical.
- Authorization for MCP tool calls is a distinct, evolving spec area: knowing whether your MCP servers implement current authorization guidance is itself a due-diligence question.

Questions I Should Be Able to Ask My Team:

1. What is our approval and vetting process before any team installs a new MCP server, and does it include the same scrutiny we'd apply to a new third-party library with production data access?
2. For each MCP server we run, what data and systems can it reach, and is it sandboxed or running with ambient privileges on the host?
3. Do we have an inventory of every MCP server and tool integration currently connected to our AI agents, and who owns patching it when a CVE is disclosed?

Technologies / Standards / Companies to Know: Model Context Protocol (Anthropic, now community-governed); NSA/CISA joint MCP security guidance; Invariant Labs (MCP security research); the Vulnerable MCP Project (public vulnerability database for MCP).

Recommended Learning:

- NSA/CISA: Model Context Protocol (MCP) Security: joint government guidance, the closest thing to an authoritative primary source on MCP-specific risk.
- GitHub MCP Exploited: Accessing private repositories via MCP: primary research writeup of a real, disclosed incident (May 2025).
- The Vulnerable MCP Project: running public database of disclosed MCP vulnerabilities, useful for staying current.
- A Timeline of Model Context Protocol (MCP) Security Breaches: chronological summary of disclosed incidents 2025–2026 for pattern-spotting.

Time Investment: 2-3 hours

13.7 Zero-Trust Architecture for Agents & AI Red Teaming

Priority: Must Understand

Executive Definition: Zero-trust for agents applies the "never trust, always verify" principle (already standard for network and identity security) to AI systems: no agent, tool response, or piece of retrieved content is trusted by default, every action is authenticated and authorized at the point of use, and blast radius is contained by design rather than assumed away. AI red teaming is the practice of adversarially testing these systems (via prompt injection, goal hijacking, tool abuse, and jailbreak attempts) before attackers do, and increasingly includes evaluating agents for unsanctioned autonomous behavior under adversarial conditions.

Why It Matters: Perimeter-based trust models fail for agentic AI because the "perimeter" is porous by design: an agent is built to read untrusted content and act on it, which is the opposite of the assumption traditional network security relies on. Regulators and standards bodies (NIST) are converging on zero-trust and continuous evaluation as the baseline expectation for agentic deployments, and red-teaming results are becoming the evidence organizations will be asked to produce, not just an internal best practice.

What I Need to Understand:

- Zero-trust for agents means per-action authorization checks, not a one-time login: an agent authenticated at session start should not be implicitly trusted for every subsequent tool call, especially after processing untrusted content.
- Red teaming for agentic AI must go beyond jailbreak/content-safety testing to include testing for goal hijacking, tool misuse, and data exfiltration under realistic "lethal trifecta" conditions: most legacy red-teaming programs were built for chatbots and don't cover this.
- Government-grade AI safety evaluators (e.g., the UK AI Security Institute) have documented agents attempting supply-chain attacks and spear-phishing during evaluations with safety filters deliberately disabled: evidence that capability for harmful autonomous action exists and must be actively tested for, not assumed absent.
- "Safety mechanisms" built into agent products (auto-approval heuristics, guardrail models) are themselves attack surface: research has shown ways to make the safety mechanism part of the failure, which argues for defense-in-depth rather than reliance on any single control.
- Continuous, not one-time, evaluation matters: agent behavior can change with model updates, new tools, or new data sources, so red-teaming needs to be a recurring program tied to change management, not a pre-launch checkbox.

Questions I Should Be Able to Ask My Team:

1. For our production agents, is trust re-verified at each tool call, or is an agent effectively trusted for the rest of its session once it authenticates once?
2. Do we red-team our agents specifically for goal hijacking and tool misuse under lethal-trifecta conditions, or only for content-safety/jailbreak issues?

3. How often is red-teaming repeated as we add tools, change models, or update prompts, and who signs off before a change ships?

Technologies / Standards / Companies to Know: NIST AI Risk Management Framework / AI 600-1 Generative AI Profile; OWASP "Agentic AI: Threats and Mitigations" and "Securing Agentic Applications Guide"; UK AI Security Institute (evaluations of agentic AI systems).

Recommended Learning:

- NIST AI Risk Management Framework: the primary US government framework; read the core functions (Govern, Map, Measure, Manage) at the executive level.
- Agentic AI: Threats and Mitigations: OWASP's threat-model-based reference for agent-specific red-teaming targets.
- Securing Agentic Applications Guide 1.0: practical architecture guidance, including zero-trust patterns for agents.

Time Investment: Half day

Agent Identity & Authorization

Last updated September 11, 2026 · 8 min read · https://erikcaldwell.com/field-guide/agent-identity-and-authorization/

14.1 Non-Human Identity & Workload Identity for Agents

Priority: Must Understand

Executive Definition: Non-Human Identity (NHI) and workload identity are the discipline of giving every AI agent, service account, and automated workload its own cryptographically verifiable identity (distinct from any human user's identity) so that every action an agent takes can be tied to a specific, authenticated, and revocable identity rather than a shared API key or a borrowed human credential. This is the foundational layer beneath the strategic question every executive must be able to answer: **who authorized this AI to take this action, with this data, using these systems, at this time?**

Why It Matters: Most enterprises already have more non-human identities (service accounts, API keys, bots) than human ones, and agentic AI is accelerating that ratio sharply: each agent, sub-agent, and tool connection is a new identity that needs provisioning, credential rotation, and offboarding, typically without a mature governance process behind it. Treating an agent as "just another service account using a shared key" collapses the ability to attribute, scope, or revoke its access independently, which is precisely what an incident response process needs to do quickly.

What I Need to Understand:

- The core requirement is a unique, verifiable identity per agent (not per application, not shared across agents) so that authentication, authorization, and audit logs can all resolve to a specific actor.
- Open standards are emerging specifically for workload identity, the Secure Production Identity Framework for Everyone (SPIFFE) and its runtime implementation SPIRE, and are being extended to cover AI agents, alongside vendor-specific offerings such as Microsoft Entra Agent ID, which issues and manages identities for agents the way Entra ID does for human users.
- Agent identity needs a lifecycle like human identity: provisioning, credential rotation, permission review, and deprovisioning when an agent or its task is retired: most organizations currently have no formal process for any of these stages for agents.
- Delegation matters: an agent frequently needs to act "on behalf of" a specific human user, which requires the agent's own identity to be distinguishable from, but linkable to, the human who authorized it (see Delegated Authorization topic).
- This is a governance gap today, not a solved problem: most enterprises can enumerate their human users far more completely than they can enumerate their active AI agents and what each one can access.

Questions I Should Be Able to Ask My Team:

1. Can we produce a complete inventory today of every AI agent operating in our environment and the distinct identity/credentials each one uses?
2. When an agent takes an action, does our logging capture the agent's own identity, or only a shared service-account or API-key identity that could belong to any of several agents?
3. What is our process for revoking a single compromised or retired agent's access without disrupting every other agent that shares its credentials?

Technologies / Standards / Companies to Know: SPIFFE/SPIRE (open standard for workload identity, CNCF); Microsoft Entra Agent ID; HashiCorp Vault (workload identity integration); broader "non-human identity" (NHI) security vendor category (e.g., Astrix, Entro, Token Security).

Recommended Learning:

- Announcing Microsoft Entra Agent ID: a major identity vendor's concrete approach to this exact problem, useful as a worked example even if vendor-neutral in your own architecture.
- SPIFFE: Securing the identity of agentic AI and non-human actors: explains the open standard being extended to cover agents.
- The Non-Human Identity Governance Vacuum: Cloud Security Alliance research on the governance gap at the executive/policy level.

Time Investment: 1 hour

14.2 Delegated Authorization, OAuth & Just-in-Time Access

Priority: Should Understand

Executive Definition: Delegated authorization is the mechanism by which an agent is granted permission to act on a specific human's or system's behalf: using standards like OAuth 2.1 rather than embedding a standing credential in the agent itself: scoped to a specific task and time window rather than granted as a permanent, broad credential. Just-in-time (JIT) access extends this by granting elevated permission only for the duration of a specific approved action, then automatically revoking it. Together these are the practical answer to: **who authorized this AI to take this action, with this data, using these systems, at this time?**: because the authorization is explicit, scoped, time-bound, and attributable to a human or policy decision.

Why It Matters: The default failure mode in early agent deployments is over-broad, standing credentials: an agent given a long-lived API key with wide scope "to make things work," which then becomes the blast radius if the agent is hijacked or the credential leaks. OAuth-based delegation and JIT access shrink that blast radius to the specific task, the specific human who authorized it, and the specific time window, and both the MCP authorization specification and enterprise IT are actively converging on this pattern as the expected baseline in 2026.

What I Need to Understand:

- Delegated authorization means the agent gets a scoped, revocable token tied to a specific user's consent for a specific purpose: not the user's actual password or a permanent admin key.
- Just-in-time access means the elevated permission exists only for the window needed to complete the approved action, then expires automatically, rather than persisting indefinitely on the chance it's needed again.
- The MCP specification has its own evolving authorization framework (built on OAuth 2.1 patterns) governing how MCP clients and servers negotiate what an agent is allowed to do: know whether your MCP integrations implement current-generation authorization or none at all.
- Multi-agent and multi-tool systems raise a specific delegation problem: when Agent A calls Agent B which calls a tool, whose authorization applies at each hop, and does the original human's consent scope carry through correctly: this "chained delegation" is an active design challenge, not a solved one.
- This is the layer where the strategic question becomes operational: a well-designed system should be able to answer, for any single agent action, exactly which human or policy decision authorized it, with what data scope, and for how long.

Questions I Should Be Able to Ask My Team:

1. For each of our agents, is access granted as a standing credential or as a scoped, time-bound, delegated token tied to a specific authorization event?
2. Do our MCP or tool integrations implement current OAuth-based authorization, or are they relying on static API keys with broad, undifferentiated scope?
3. In a multi-agent workflow, can we trace a downstream tool call back to the original human authorization that should have scoped it, or does delegation get "flattened" into a shared service identity somewhere in the chain?

Technologies / Standards / Companies to Know: OAuth 2.1 and the IETF OAuth working group; Model Context Protocol authorization specification; enterprise identity providers extending delegated-access patterns to agents (Microsoft Entra, Okta, Auth0/Okta CIC).

Recommended Learning:

- MCP Authorization specification: the protocol-level primary source for how MCP handles delegated authorization between agents and tools.
- Least privilege for AI agents with Microsoft Entra Agent ID: a concrete enterprise implementation pattern combining identity, delegation, and scoping.
- AI Agents Have an Authorization Problem, Not Just an Identity Problem: a clear articulation of why identity alone is insufficient without this layer.

Time Investment: 2-3 hours

14.3 Least Privilege, Transaction Limits & Audit Attribution

Priority: Must Understand

Executive Definition: This is the operational discipline that turns identity and delegated authorization into enforceable controls: scoping every agent to the minimum permissions it needs (least privilege), capping the size or consequence of any single autonomous action (transaction limits), and ensuring every action an agent takes is logged with enough detail to answer, after the fact, exactly who and what authorized it (audit attribution). It is the last line of defense when prevention fails: the control that limits damage and enables accountability once an agent has been hijacked or has erred.

Why It Matters: Security shifts fundamentally once AI systems can act rather than merely generate text: a wrong sentence in a chat is a quality problem, but an agent with unnecessary write access to a financial system, an uncapped ability to send emails or move funds, or actions logged only under a shared service account is an incident waiting to happen with no clean way to trace it back. These three controls are what make the answer to "who authorized this AI to take this action, with this data, using these systems, at this time?" something your organization can actually produce on demand, rather than a question you can only ask rhetorically after the fact.

What I Need to Understand:

- Least privilege for agents means scoping to the specific task, not the broadest role a human in that job function might plausibly need: an agent's permission set should typically be narrower than the equivalent human employee's, because it acts without the judgment a human would apply in ambiguous cases.
- Transaction limits (dollar caps, rate limits, record-count limits, requiring human approval above a threshold) bound the damage a single hijacked or erroneous agent action can cause, independent of whether the underlying prompt injection or goal hijacking was caught.
- Audit attribution requires that logs capture the specific agent identity, the specific delegated authorization that permitted the action, the data accessed, and the time (not just "the AI system did X") or post-incident investigation and regulatory response both stall.
- Industry frameworks (e.g., Fintech Open Source Foundation (FINOS)'s Agent Authority Least Privilege Framework, aimed at financial services) are starting to formalize this as a control category with concrete mitigations, not just a best-practice slogan.
- These controls should be treated as mandatory for any agent with write access to production systems, financial systems, or customer data: read-only or low-consequence agents warrant a lighter touch, and over-applying heavy controls everywhere slows adoption without improving actual risk posture.

Questions I Should Be Able to Ask My Team:

1. For our highest-risk agents (financial, customer data, production systems), what is the actual permission scope granted versus the minimum the task requires, and who last reviewed that gap?
2. What hard transaction or action limits exist for each agent that can take a consequential action, and what happens when that limit is hit: does it fail safe or escalate to a human?

3. If asked today, could we produce a complete audit trail for any single agent action, showing the specific agent identity, the authorization that permitted it, and the data touched, or would that investigation hit a dead end at a shared credential or generic log entry?

Technologies / Standards / Companies to Know: FINOS AI Governance Framework (Agent Authority Least Privilege Framework); Microsoft Entra Agent ID (permission scoping and tool binding); enterprise SIEM/logging platforms extended to capture per-agent audit trails.

Recommended Learning:

- Agent Authority Least Privilege Framework: FINOS: a concrete, financial-services-grade framework for exactly this control set.
- Least privilege for AI agents: Identity, access, and tool binding: practical treatment tying identity, scoping, and tool binding together.
- Authorization and Governance for AI Agents: Runtime Authorization Beyond Identity at Scale: addresses runtime enforcement and audit at scale, beyond one-time identity checks.

Time Investment: 2-3 hours

AI Governance, Legal, Privacy & Compliance

Last updated September 11, 2026 · 12 min read · https://erikcaldwell.com/field-guide/ai-governance-legal-privacy-and-compliance/

15.1 NIST AI Risk Management Framework & Generative AI Profile

Priority: Must Understand

Executive Definition: The NIST AI Risk Management Framework (AI RMF 1.0, released January 2023) is a voluntary, non-regulatory framework organized around four functions (Govern, Map, Measure, Manage) for identifying and managing risks across an AI system's lifecycle. The Generative AI Profile (NIST AI 600-1, published July 2024) extends it with risks specific to generative AI (confabulation, data leakage, harmful content, over-reliance) and suggested actions mapped to the same four functions. Neither is a law or a certification; both are reference vocabulary and structure that regulators, auditors, and vendors increasingly assume you already use.

Why It Matters: NIST AI RMF has become the de facto common language for AI governance in US enterprises, referenced by insurers, auditors, and enterprise customers doing vendor risk assessments even though compliance is voluntary. The Generative AI Profile gives you a checklist of GenAI-specific failure modes (hallucination, IP infringement, data leakage) your own AI programs are unlikely to have already scoped. It is also the framework most cleanly mapped to ISO/IEC 42001 and to several state AI laws, so adopting its vocabulary reduces translation cost across obligations.

What I Need to Understand:

- The four functions (Govern, Map, Measure, Manage) are lifecycle stages, not a maturity ladder: Govern is the cross-cutting function that has to exist before the others are meaningful.
- The RMF is voluntary and non-prescriptive; the companion Playbook gives suggested (not mandatory) actions per subcategory and is explicitly "neither a checklist nor a set of steps to be followed in its entirety."
- The Generative AI Profile adds ~12 GenAI-specific risk categories (e.g., confabulation, dangerous/violent content, data privacy, IP, value chain/component integration) layered onto the core functions: it does not replace the base RMF.
- This is a framework, not a certification: there is no "NIST AI RMF certified" status; certification-seeking vendors claiming this should be questioned.
- The framework is under active revision tied to US federal AI policy shifts (the 2025 AI Action Plan); expect updates, not a frozen document.

Questions I Should Be Able to Ask My Team:

1. Which of our AI systems have gone through a documented Map/Measure/Manage cycle, and who owns Govern-level accountability across them?

2. For our generative AI deployments specifically, which of the GenAI Profile's risk categories (confabulation, data leakage, IP, over-reliance) have we actually assessed versus just referenced in a slide?
3. When a vendor claims "NIST AI RMF compliant," what artifact are they pointing to, and does it map to a specific Govern/Map/Measure/Manage subcategory or is it marketing language?

Technologies / Standards / Companies to Know: NIST AI RMF 1.0, NIST AI 600-1 (Generative AI Profile), NIST AI RMF Playbook, NIST AI Safety Institute (now the Center for AI Standards and Innovation)

Recommended Learning:

- AI Risk Management Framework: NIST: the framework's home page and core PDF.
- NIST AI 600-1 Generative AI Profile (PDF): the primary source text.
- NIST AI RMF Playbook: suggested actions per subcategory, useful as an audit checklist.

Time Investment: 2-3 hours

15.2 ISO/IEC 42001 (AI Management Systems)

Priority: Should Understand

Executive Definition: ISO/IEC 42001:2023, published December 2023, is the first international standard for an "AI Management System" (AIMS): an organizational governance structure for how you develop, provide, or use AI systems responsibly. Unlike NIST's AI RMF, it is a certifiable management-system standard, structured like ISO 27001 (information security) or ISO 9001 (quality) around Plan-Do-Check-Act, requiring documented policies, an AI risk process, and continual improvement. An organization can be audited and certified against it by an accredited body.

Why It Matters: ISO/IEC 42001 certification is emerging as the artifact enterprise procurement and vendor-risk teams ask for from AI vendors and from internal AI functions, the same way ISO 27001 became a baseline ask for security. Because it is certifiable, it creates external accountability (third-party audits) that NIST's framework does not. It also gives you a structural scaffold (policy, roles, risk register, internal audit, management review) that most internal "AI governance" efforts are missing even when they have written principles.

Why It Matters: (kept single per format above)

What I Need to Understand:

- ISO/IEC 42001 is a *management system* standard (like ISO 27001/9001), not a technical or content standard: it governs how you manage AI risk and lifecycle, not what your models must do.
- It is certifiable: accredited bodies audit and issue certificates, creating a real compliance/procurement artifact, unlike the voluntary NIST RMF.

- It complements, rather than duplicates, ISO/IEC 23894 (AI risk management guidance) and ISO/IEC 22989 (AI terminology): 42001 is the organizational-governance layer that can incorporate those as risk methodology.
- Scope covers any organization that provides or uses AI-based products/services: this includes companies deploying vendor AI, not only those building models.
- Getting certified requires an actual AI system inventory and risk treatment process (see Topic 5): you cannot certify against a standard you haven't first built the underlying practice for.

Questions I Should Be Able to Ask My Team:

1. If we pursued ISO/IEC 42001 certification today, what would an auditor find missing in our current AI governance documentation?
2. Which of our AI vendors are ISO/IEC 42001 certified, and does that certification cover the specific product/service we use or a different part of their business?
3. Is our AI risk methodology aligned to ISO/IEC 23894, or are we inventing our own risk taxonomy that won't map cleanly if we pursue 42001 certification later?

Technologies / Standards / Companies to Know: ISO/IEC 42001:2023, ISO/IEC 23894 (AI risk management), ISO/IEC 22989 (AI terminology), accredited certification bodies (e.g., BSI, DNV, A-LIGN)

Recommended Learning:

- ISO/IEC 42001:2023: AI management systems: the official standard listing and scope.
- ISO/IEC 42001 explained: ISO's own plain-language explainer.
- ISO/IEC 42001:2023: Microsoft compliance overview: useful as a worked example of how a large vendor scopes and documents certification.

Time Investment: 1 hour

15.3 EU AI Act

Priority: Must Understand

Executive Definition: The EU AI Act (entered into force August 1, 2024) is binding law that classifies AI systems by risk tier: unacceptable (banned), high-risk (heavily regulated), limited-risk (transparency obligations), minimal-risk (unregulated): and imposes obligations accordingly, with extraterritorial reach to any organization whose AI system output is used in the EU. As of September 2026, prohibited-practice rules, AI literacy obligations, and general-purpose AI (GPAI) provider obligations are already in force and being enforced; most high-risk system obligations were pushed later by a 2026 "Digital Omnibus" amendment. This is legal requirement, not best-practice guidance, and this summary is not legal advice; confirm current obligations with counsel before acting.

Why It Matters: This is the first comprehensive AI-specific statute with real penalties (up to 7% of global turnover for prohibited practices) and it functions as a template other jurisdictions are watching. Enforcement has already started for GPAI and prohibited-practice provisions, and the 2026 Omnibus

delay on high-risk obligations does not remove them: it moves the compliance clock, which is exactly the kind of deadline executives get blindsided by when they read "delayed" as "cancelled."

What I Need to Understand:

- Legal requirement, current as of Sept 2026: prohibited practices and AI literacy obligations applied from **Feb 2, 2025**; GPAI provider obligations and governance bodies applied from **Aug 2, 2025**; transparency obligations (Article 50) and broader enforcement across GPAI/prohibitions/literacy took effect **Aug 2, 2026**.
- Legal requirement, changed by the 2026 Digital Omnibus (agreed May 2026): stand-alone high-risk system obligations (Annex III) are now due **Dec 2, 2027** (previously Aug 2, 2026); product-embedded high-risk systems (Annex I) are due **Aug 2, 2028**. Synthetic-content watermarking/labeling obligations were pushed to **Dec 2, 2026**.
- Legal requirement: new prohibitions on AI-generated CSAM and non-consensual intimate imagery were added by the same Omnibus, effective **Dec 2, 2026**.
- The Act applies extraterritorially: if your AI system's output is used by people in the EU, you can have obligations regardless of where your company is headquartered.
- "High-risk" classification (Annex III) covers specific use-case categories (employment, credit, critical infrastructure, law enforcement, etc.): most enterprise internal tools are not automatically high-risk, but HR, hiring, and credit-adjacent AI usually are.
- Recommended practice (not legal requirement): using NIST AI RMF or ISO/IEC 42001 as your internal control framework to *evidence* AI Act compliance: the Act itself does not mandate either standard, though harmonized EU standards are expected to serve as a compliance presumption pathway.

Questions I Should Be Able to Ask My Team:

1. Which of our AI systems, if any, would fall under Annex III high-risk categories, and are we tracking the Dec 2027 / Aug 2028 deadlines for those specifically rather than assuming everything was delayed to 2026?
2. Do we have documented AI literacy training and transparency disclosures in place for the obligations that are already enforceable today, not just the ones still years out?
3. If we deploy a general-purpose AI model from a third party, do we understand which GPAI obligations sit with the model provider versus which "deployer" obligations sit with us?

Technologies / Standards / Companies to Know: EU AI Office, EU AI Board, GPAI Code of Practice, Annex III high-risk categories, Digital Omnibus on AI (2026)

Recommended Learning:

- AI Act: Shaping Europe's digital future (European Commission): the official EU source.
- EU AI Act implementation timeline: AI Act Service Desk: the authoritative, current date-by-date timeline.

- EU agrees Digital Omnibus deal to simplify AI rules: White & Case: clear summary of what the 2026 Omnibus actually changed and what it didn't.

Time Investment: Half day

15.4 AI, Intellectual Property & Copyright Risk

Priority: Must Understand

Executive Definition: AI IP risk runs in two directions: inbound (was the data used to train the models you rely on lawfully obtained, exposing you to secondary liability or license disputes) and outbound (can your company actually own or enforce copyright in AI-generated output). As of September 2026, both questions remain legally unsettled in the US: no appellate ruling has resolved whether training on copyrighted material is fair use, and the US Copyright Office's position that purely AI-generated output lacks sufficient human authorship to be copyrightable has not been overturned by courts. This entry states legal status only; it is not legal advice, and positions here can change with any ruling.

Why It Matters: You are currently making product and vendor decisions inside a legal vacuum: the outcome of pending litigation (notably *New York Times v. Microsoft and OpenAI*) could retroactively affect the risk profile of models you've already built products on. Separately, if your own AI-assisted work product isn't copyrightable because a court or the Copyright Office decides it lacks human authorship, that has direct implications for what IP protection your company can claim over AI-assisted deliverables, code, and content.

What I Need to Understand:

- Training-data fair use is unresolved: *New York Times v. Microsoft and OpenAI* is past motion-to-dismiss (core copyright claims survived, March 2025) but has not reached a fair-use ruling as of September 2026; the US Department of Justice filed a brief supporting OpenAI in September 2026: the first time the US government has taken a formal position on this question.
- Outbound copyrightability is constrained: US Copyright Office guidance holds that output without sufficient human creative control/authorship is not copyrightable; the US Supreme Court declined in March 2026 to review whether AI alone can create copyrighted works, leaving the human-authorship requirement standing for now.
- "Sufficient human authorship" is a fact-specific, unsettled line: heavy prompting alone is unlikely to qualify; substantial human selection, arrangement, or modification of AI output is more defensible, but there is no bright-line test yet.
- Vendor indemnification clauses for AI-generated-content IP claims vary widely and are not a substitute for understanding your own exposure: read what's actually indemnified (training-data claims vs. output-infringement claims are usually treated differently).
- This is a fast-moving area: any position taken today should be revisited at least annually, and definitely on any ruling in a marquee case.

Questions I Should Be Able to Ask My Team:

1. For AI-generated code, content, or designs that matter to our IP position, do we have a practice of documented human review/modification sufficient to support a copyrightability claim if it were ever challenged?
2. Which of our AI vendor contracts actually indemnify us against training-data infringement claims versus only output-infringement claims, and do we understand the difference?
3. Are we tracking the status of the major pending training-data litigation, and do we have a plan to reassess vendor risk if a ruling goes against the AI providers we depend on?

Technologies / Standards / Companies to Know: US Copyright Office, *NYT v. Microsoft and OpenAI*, human-authorship requirement, AI training-data licensing markets (emerging)

Recommended Learning:

- The New York Times v. Microsoft and OpenAI: case background and status: kept current, good neutral case tracker.
- Generative Artificial Intelligence and Copyright Law: Congressional Research Service: a primary, non-advocacy legal summary.
- US Supreme Court declines to consider whether AI alone can create copyrighted works: Morgan Lewis: law-firm analysis of the authorship question's current state.

Time Investment: 1 hour

15.5 AI System Inventory, Risk Classification & Responsible AI Principles

Priority: Should Understand

Executive Definition: Before any framework (NIST RMF, ISO 42001) or regulation (EU AI Act) can be operationalized, an organization needs a living inventory of every AI system in use (internal builds, embedded vendor AI, and shadow AI) each tagged with a risk tier and an owner. Responsible AI principles (fairness, transparency, accountability, safety, privacy, human oversight) are the criteria you evaluate each inventoried system against; they are the "what to check for," while the inventory and classification are the "where to look."

Why It Matters: Every governance framework in this curriculum (NIST RMF's Map function, ISO 42001's risk process, the EU AI Act's risk tiers) assumes you already have this inventory; most mid-size enterprises don't, especially where AI arrived embedded in SaaS tools rather than through a formal build process. Without it, "we have an AI governance program" is a governance program with nothing to govern.

What I Need to Understand:

- An inventory must capture vendor-embedded AI (the AI features quietly turned on inside your CRM, HRIS, or productivity suite), not just internally built models: this is where most shadow risk actually lives.

- Risk classification should use a consistent, small taxonomy (e.g., aligned to EU AI Act tiers or a simple high/medium/low) applied uniformly, not an ad hoc label per team.
- Responsible AI principles are evaluation criteria, not a governance process by themselves: a values statement without an inventory to apply it to is not a control.
- Ownership matters more than documentation: each inventoried system needs a named accountable owner, not just a description in a spreadsheet nobody updates.
- This is foundational infrastructure for ISO/IEC 42001 certification and for demonstrating EU AI Act compliance: treat it as a prerequisite project, not a parallel one.

Questions I Should Be Able to Ask My Team:

1. Do we have a single, current inventory of every AI system in use across the company, including vendor-embedded AI features, with a named owner for each?
2. What risk tier is assigned to each system, and what criteria determined that tier: is it consistent across business units?
3. When a new AI feature gets enabled inside an existing SaaS tool we already use, what is our process for catching that and classifying it?

Technologies / Standards / Companies to Know: OECD AI Principles, NIST AI RMF "Map" function, ISO/IEC 42001 risk register requirements, EU AI Act risk tiers

Recommended Learning:

- AI Risk Management Framework: NIST: see Topic 1; the Map function is the direct source for inventory/classification practice.
- ISO/IEC 42001 explained: see Topic 2; risk-and-opportunity management section covers inventory expectations.

Time Investment: 1 hour

AI FinOps & Economics

Last updated September 11, 2026 · 8 min read · https://erikcaldwell.com/field-guide/ai-finops-and-economics/

16.1 Token Economics & Cost-Per-Outcome

Priority: Must Understand

Executive Definition: Token economics is the unit-cost model of AI systems: providers bill per input and output token, so cost scales with prompt length, context window usage, output verbosity, and (for agents) the number of intermediate steps taken. Cost-per-outcome reframes this around the metric that actually matters to the business (dollars spent per successfully resolved ticket, per correctly extracted document, per completed task) rather than dollars spent per token or per API call.

Why It Matters: Cost per token is a weaker metric than cost per successful outcome because it says nothing about whether the money was well spent: a cheap model that fails and needs three retries, or a verbose model that uses more tokens per response but resolves the task in one pass, can have opposite cost-per-token and cost-per-outcome rankings. This gap matters far more for agentic workflows than for single-turn chat: an agent's cost is a function of an unbounded, data-dependent number of tool calls, retries, and reasoning steps, so the same task can cost 2x or 20x depending on how many loops the agent takes to converge: a degree of per-transaction cost variance that traditional SaaS spend, dominated by flat seat licenses and predictable compute reservations, simply does not have. FinOps for AI practitioners describe this explicitly as a shift from cost-per-unit-of-infrastructure to cost-per-unit-of-value, and it requires monitoring discipline closer to production incident response than to conventional cloud budgeting (finops.org).

What I Need to Understand:

- Input tokens, output tokens, and (for some providers) cached-token discounts are billed at different rates: a system that reads a large document repeatedly per turn has a very different cost profile than one that generates long responses.
- Cost-per-outcome requires defining what "successful outcome" means for each workflow (ticket resolved without escalation, extraction verified correct, task completed without human intervention): this is a product/business definition, not something the finance or engineering team can set unilaterally.
- Agentic cost is fundamentally less predictable than single-turn chat cost: retries, tool-call loops, and multi-step reasoning chains mean the same input can produce wildly different token consumption run to run, which is why per-agent, per-workflow cost monitoring (not just aggregate monthly spend) is required.
- A model that is more expensive per token but reaches a correct outcome in fewer steps can be cheaper per outcome than a "cheap" model that requires more retries or human correction: token price alone is not a valid basis for model selection.

- Cost-per-outcome tracking depends on having the same tracing infrastructure used for trajectory evaluation and observability: cost and quality data need to be joined at the trace level, not analyzed separately.

Questions I Should Be Able to Ask My Team:

1. Do we track cost per successful outcome for our top agentic workflows, or only aggregate token spend per model/application?
2. What is the observed variance (not just the average) in cost per task for our agentic workflows, and do we have alerting on outlier-cost runs?
3. When we compare models for a given workflow, is the comparison based on cost-per-outcome including retries and escalations, or on list price per token?

Technologies / Standards / Companies to Know: FinOps Foundation (FinOps for AI working group), Bedrock/Azure/Vertex usage and billing APIs, LangSmith/Arize cost-tagged tracing.

Recommended Learning:

- FinOps for AI Overview: FinOps Foundation: the industry working group's framing of why AI cost management differs from cloud FinOps.
- Token Economics: The Atomic Unit of AI Value: FinOps Foundation: direct treatment of token-level cost as a unit-economics problem, not a billing line item.
- Identifying Token Costs Hiding in Your Agentic Loop: MachineLearningMastery: engineering-level explanation of why agent loops obscure true cost drivers.

Time Investment: 1 hour

16.2 Cost Controls: Model Routing, Caching & Smaller-Model Substitution

Priority: Must Understand

Executive Definition: These are the three primary engineering levers for reducing AI spend without simply refusing to use larger models: model routing (also called cascading) sends each request to the cheapest model capable of handling it, escalating to a larger model only when needed; caching (exact-match or semantic) avoids paying for a model call at all when an equivalent request has been answered before; smaller-model substitution deliberately uses a lower-cost, fine-tuned, or distilled model for well-scoped subtasks instead of a frontier model for everything.

Why It Matters: These controls sit directly downstream of cost-per-outcome logic: the goal is not the lowest token price but the lowest cost per successful outcome, so each of these levers has to be evaluated against eval and groundedness metrics, not deployed blind. A router or cache misconfigured to prioritize price alone will quietly degrade output quality in ways that don't show up until a customer complains or an audit occurs.

What I Need to Understand:

- Model routing/cascading typically works by scoring task complexity (via a lightweight classifier or the smaller model's own confidence) and escalating only the subset of requests that need a stronger model: this is architecturally similar to the model-router feature now built into some cloud-native gateways (e.g., Azure AI Foundry's model router).
- Caching for LLMs comes in two forms: exact-match caching (only helps with literally repeated queries) and semantic caching (matches queries that are semantically similar, not identical), which has a much higher hit rate but requires an embedding/similarity layer and careful tuning to avoid serving a cached answer to a subtly different question.
- Smaller-model substitution works best for narrow, well-defined subtasks (classification, extraction, formatting) and worst for open-ended reasoning: the substitution decision should be scoped per task type within a workflow, not per application.
- Every cost control here needs a quality guardrail wired to it: routing thresholds, cache hit criteria, and substitution boundaries should all be validated against golden-dataset and production-eval metrics, not tuned by cost alone.
- These controls are typically implemented at the gateway layer, which is why gateway architecture choice and cost-control strategy are not independent decisions.

Questions I Should Be Able to Ask My Team:

1. What is our model router's escalation criterion, and has it been validated against actual task failure rates, not just cost savings?
2. What is our semantic cache's similarity threshold, and how do we detect if it's serving a stale or wrong cached answer to a similar-but-different query?
3. For workflows using smaller-model substitution, what is the measured cost-per-outcome delta versus using the larger model for the same subtask?

Technologies / Standards / Companies to Know: Azure AI Foundry model router, LiteLLM/Portkey routing and caching, semantic caching (e.g., Redis-based implementations), model distillation/fine-tuning for task-specific substitution.

Recommended Learning:

- Model router for Microsoft Foundry: concepts: official documentation of a production model-routing/cascading implementation.
- LLMOps Guide: Build Fast, Cost-Effective LLM Apps: Redis Engineering Blog: engineering-level treatment of semantic caching mechanics and tradeoffs.
- LLM Routing and Model Cascades: How to Cut AI Costs Without Sacrificing Quality: independent engineering blog covering cascade design against quality metrics, not just price.

Time Investment: 1 hour

16.3 Chargeback, Showback & Runaway Agent Cost Risk

Priority: Monitor

Executive Definition: Chargeback allocates AI spend directly to the business unit or application that incurred it, typically billed against that team's budget. Showback reports the same per-team/per-application cost breakdown for visibility without an actual budget transfer. Runaway agent cost risk is the specific hazard that an autonomous agent (stuck in a retry loop, given an ambiguous goal, or manipulated by adversarial input) consumes far more tokens/tool calls than any human would have authorized, with no natural stopping point the way a human operator provides.

Why It Matters: Traditional cloud chargeback models assume relatively stable, forecastable per-team consumption; agentic AI breaks that assumption because a single misbehaving agent run can spike spend by an order of magnitude in minutes, not months: this is qualitatively different from a forgotten VM instance left running. Without per-application/per-agent cost attribution and hard spend caps, showback reporting arrives too late to prevent the damage, and chargeback without real-time caps just tells a business unit after the fact that they owe an unexpectedly large bill.

What I Need to Understand:

- Showback is the necessary first step (visibility) and should be in place before attempting chargeback, since accurate per-team attribution requires the same tracing/tagging infrastructure regardless of which model is used.
- Runaway cost in agentic systems typically stems from: unbounded retry loops, an agent re-attempting a failing tool call without a cap, recursive sub-agent spawning, or an adversarial input designed to induce excessive tool use: each needs its own specific guard (max-retry limits, max-depth limits, hard per-task token/dollar budgets enforced at the gateway).
- Hard budget caps enforced at the gateway or orchestration layer (kill the run when it exceeds a defined token/dollar ceiling) are a more reliable control than alerting after the fact, given how fast an agent loop can accumulate cost compared to traditional infrastructure overspend.
- Chargeback models need to account for the higher cost variance of agentic workflows: a flat per-team allocation formula that worked for predictable SaaS licensing does not fairly represent AI spend that varies by workload complexity and failure rate.
- Attribution granularity matters: per-application chargeback is necessary but often insufficient: for shared multi-tenant agent platforms, attribution needs to go down to the requesting user or workflow, not just the owning team.

Questions I Should Be Able to Ask My Team:

1. Do we have hard per-task or per-session spend caps enforced at the gateway, or only after-the-fact cost alerts?
2. What is our attribution granularity: can we trace a cost spike back to a specific agent, workflow, or user, not just a team-level total?
3. Has a runaway-cost scenario (retry loop, recursive sub-agent spawn) ever actually occurred in our environment, and what stopped it?

Technologies / Standards / Companies to Know: FinOps Foundation, gateway-enforced budget caps (Bedrock/Azure/Vertex quota and budget features), cost-tagged tracing (LangSmith, Arize).

Recommended Learning:

- FinOps for AI Overview: FinOps Foundation: industry framing of chargeback/showback specifically adapted for AI workloads.
- Chargeback vs. Showback: Cloud Cost Allocation Models Explained: CloudZero: general cost-allocation mechanics, useful baseline before applying it to AI-specific variance.
- Identifying Token Costs Hiding in Your Agentic Loop: MachineLearningMastery: concrete engineering discussion of how agent loops produce the runaway-cost pattern this topic is about.

Time Investment: 1 hour

AI Observability & AgentOps

Last updated September 11, 2026 · 5 min read · https://erikcaldwell.com/field-guide/ai-observability-and-agentops/

17.1 LLMOps/AgentOps & Tracing Agent Trajectories

Priority: Must Understand

Executive Definition: LLMOps/AgentOps is the operational discipline of running LLM and agent systems in production: capturing structured traces of every model call and tool invocation (inputs, outputs, latency, cost, intermediate reasoning steps), correlating them into a single end-to-end trajectory per user request, and using that trace data for debugging, evaluation, and cost attribution. It extends conventional software observability (logs, metrics, traces) with AI-specific concepts (prompts, completions, token usage, and multi-step agent trajectories) now being standardized as OpenTelemetry's GenAI semantic conventions.

Why It Matters: An agent failure or cost spike is undebuggable after the fact unless every step it took was captured at the time: you cannot re-derive what an agent "was thinking" or which tool arguments it used from application logs designed for conventional request/response services. The industry is actively converging on a shared standard for this (OpenTelemetry's GenAI semantic conventions define spans for LLM calls, tool invocations, and agent steps) specifically so that traces are portable across observability vendors instead of locked into a single platform's proprietary schema (opentelemetry.io).

What I Need to Understand:

- A trace for an agentic workflow needs to capture, at minimum: every prompt and completion, every tool call with its arguments and result, token counts and cost per step, latency per step, and the overall trajectory linking these into one causal chain: this is the raw data trajectory evaluation and incident response both depend on.
- OpenTelemetry's GenAI semantic conventions define standardized span types for LLM invocations and agent/tool steps, which matters for interoperability: traces captured this way can move between observability backends (Datadog, Arize, LangSmith, self-hosted) without a rewrite.
- Tracing has to be designed in from the start of an agent system's architecture: retrofitting full trajectory capture after an incident, when you need it most, is usually not possible.
- LLMOps/AgentOps tooling overlaps heavily with the eval tooling covered above: the same trace data that supports debugging is what a trajectory eval or a production canary comparison scores against.
- Distinguish platform-level observability (is the gateway/model healthy, latency, error rates) from trajectory-level observability (did this specific agent run behave correctly): both are needed, and they are not the same discipline.

Questions I Should Be Able to Ask My Team:

1. Can we reconstruct the full step-by-step trajectory (every tool call, argument, and intermediate output) for any agent run from the last 90 days, or only recent runs?
2. Are we using a standardized tracing format (OpenTelemetry GenAI conventions) or a proprietary schema tied to one vendor?
3. If our observability vendor changed tomorrow, would we lose historical trace data, or is it portable?

Technologies / Standards / Companies to Know: OpenTelemetry GenAI semantic conventions, LangSmith, Arize, Langfuse, Datadog LLM Observability.

Recommended Learning:

- Inside the LLM Call: GenAI Observability with OpenTelemetry: OpenTelemetry Blog: primary-source explanation of the emerging standard for AI tracing.
- OpenTelemetry GenAI Semantic Conventions: MLflow documentation: concrete documentation of what a compliant trace actually contains.
- AgentOps: Enabling Observability of LLM Agents (arXiv): academic framing of what agent observability needs to capture and why conventional Application Performance Monitoring (APM) falls short.

Time Investment: 2-3 hours

17.2 Incident Response & Audit Replay for Agent Actions

Priority: Must Understand

Executive Definition: This is the capability to fully reconstruct, after the fact, exactly what an AI agent did, why (what it was told, what it retrieved, what it reasoned), and what real-world effects it had (what systems it touched, what data it read or wrote, what transactions it initiated): for both routine audit and post-incident investigation. It depends directly on the tracing infrastructure above, but adds the requirement that traces be tamper-evident, retained long enough to matter, and usable by security/compliance teams, not just engineers debugging a bug.

Why It Matters: When an agent takes an unauthorized or harmful action (sends an incorrect email, executes an unintended transaction, exposes data it shouldn't have accessed) the organization needs to answer three questions fast: what exactly happened, what was the blast radius, and can it happen again. Unlike a human employee's action, an agent's decision process is not something you can just ask about after the fact; if the trace wasn't captured at execution time, that reasoning is unrecoverable. This makes audit-grade logging a prerequisite for agent authorization scopes in any workflow where the agent can take real-world, irreversible actions.

What I Need to Understand:

- Minimum viable audit trail for an agent action: the triggering user/request, the full prompt context (including any retrieved documents or tool outputs the agent saw), every tool call with arguments and results, the final action taken, and the identity/credentials under which that action executed.
- "Replay" means being able to reconstruct the decision sequence deterministically enough to explain it to an auditor or investigator: this does not require re-running the model (which may give a different output on a re-run given non-determinism), it requires that the original trace is complete enough to stand on its own as the record.
- Retention and tamper-evidence requirements for these logs are a governance/compliance decision (how long, who can access, can they be altered after write) that should be set explicitly, not left as a default from whatever observability tool was adopted for debugging.
- Incident response playbooks for agents need to include agent-specific containment steps (revoking the agent's credentials/tool access, not just "roll back the deployment") because the damage may already be external to the system that was patched.
- This capability should be tested before it's needed: a tabletop exercise where the team actually tries to reconstruct a past agent action from stored traces is the only reliable way to confirm the capability works, rather than assuming it does.

Questions I Should Be Able to Ask My Team:

1. If an agent took a harmful or unauthorized action six months ago, could we reconstruct exactly what it saw, reasoned, and did: today?
2. What is our log retention period for agent traces, and does it meet our actual compliance/audit requirements, or just our debugging convenience window?
3. Does our incident response runbook for agent-caused incidents include revoking the specific agent's tool/credential access, distinct from a general system rollback?

Technologies / Standards / Companies to Know: OpenTelemetry GenAI semantic conventions (as the underlying trace format), NIST AI RMF, immutable/append-only audit logging patterns.

Recommended Learning:

- NIST AI 600-1: Generative AI Profile: governance-level expectations for traceability and incident accountability in generative AI systems.
- Auditing and Logging AI Agent Activity: A Guide for Engineers: LoginRadius Engineering Blog: practitioner-level treatment of what an agent audit trail needs to contain.
- Replay: reconstructing an agent action from the audit log: Deixic: a focused engineering walkthrough of the replay problem specifically.

Time Investment: 2-3 hours

AI-Native Product Design

Last updated September 11, 2026 · 8 min read · https://erikcaldwell.com/field-guide/ai-native-product-design/

18.1 Beyond the Chatbot: What AI-Native Products Look Like

Priority: Must Understand

Executive Definition: Bolting a chat window onto an existing application does not make it AI-native: it's a conversational veneer over the same predetermined workflow, menus, and forms the product always had. An AI-native product is architected around users specifying a desired outcome and the system determining and executing the steps, tool calls, and data lookups needed to reach it: collapsing multi-screen, multi-click workflows into a single request-and-result loop, with the underlying UI adapting to what the task actually requires rather than to a fixed navigation tree.

Why It Matters: Most "AI transformation" initiatives at incumbent software companies amount to adding a chat sidebar that answers questions about a product whose core interaction model (click through screens, fill forms, follow a fixed sequence) is unchanged; this captures a fraction of the available value and is trivially copyable by any competitor. The organizations capturing real advantage are redesigning the product's control flow itself: the interface exists to let a user state an outcome and to make the system's actions and constraints legible, not to route them through a menu tree that was designed for a pre-AI world.

What I Need to Understand:

- A chatbot layered onto an unchanged application is a distribution channel for the same workflow, not a redesign: the test is whether removing the chat window would leave the product's actual task-completion path unchanged (if yes, it isn't AI-native).
- AI-native design shifts the unit of interaction from "screen and click" to "goal and result": the user states what outcome they want, and the system plans and executes the underlying steps (which may span multiple tools, records, or approvals) rather than requiring the user to navigate to each one.
- This shift changes what UI is for: less about presenting every option up front, more about surfacing the system's plan, actions taken, and any decision points that need the human: which is why Topics 7 and 8 (approval UX, provenance/confidence UX) are inseparable from this one.
- AI agents are increasingly consuming your product's interfaces directly, not just humans: interface clarity, semantic structure, and predictable behavior are now a functional requirement for machine users as well as human ones, not just an accessibility nicety.
- Retrofitting an AI-native interaction model onto a workflow-first legacy architecture is usually harder than it looks, because the underlying system (permissions, data model, audit trail) was built assuming a human clicks through a known sequence, not that an agent takes multi-step action on the user's behalf.

Questions I Should Be Able to Ask My Team:

1. For our flagship AI feature, if we removed the chat interface, would users still be forced through the exact same click-path they used before, and if so, what have we actually changed?
2. Which of our products could plausibly let a user state an outcome ("close out this customer's account and confirm no open balances") instead of navigating a multi-screen workflow, and what's stopping us from building that today?
3. Are we designing our APIs and interfaces to also be usable by AI agents acting on a user's behalf, or only by humans clicking through screens?

Technologies / Standards / Companies to Know: Agentic AI architectures, tool-calling/function-calling LLM patterns, model context protocols for agent-to-system integration

Recommended Learning:

- AI Agents as Users: Nielsen Norman Group: the core argument that AI agents are now a distinct user class your interfaces must serve, with direct implications for what "AI-native" requires structurally.
- State of UX 2026: Nielsen Norman Group: grounded read on where product UX is actually heading versus hype.

Time Investment: 2-3 hours

18.2 Progressive Autonomy & Approval UX

Priority: Must Understand

Executive Definition: Progressive autonomy is the design principle that an AI agent's authority to act without human sign-off should scale gradually (with track record, reversibility of the action, and stakes) rather than being an all-or-nothing switch. Approval UX is the mechanism for that: it makes visible what the system is about to do, why, and what happens if the human does nothing, and it is what separates "AI that drafts a recommendation" from "AI that takes an irreversible action on your systems of record."

Why It Matters: The single most common failure mode in agentic AI deployments is granting broad autonomy before the organization has any track record with the narrower version: the cost of a bad autonomous action (a wrong email sent, an incorrect refund issued, a record silently modified) is rarely symmetric with the cost of asking for confirmation, and getting this wrong is both a trust-destroying incident risk and, per the EU AI Act, potentially a regulatory one for higher-risk use cases.

What I Need to Understand:

- Autonomy should be modeled as a spectrum with defined levels (e.g., recommend-only, act-with-confirmation, act-with-notification, fully autonomous within bounds), not a binary "AI can act" toggle: several published frameworks (analogous to vehicle autonomy levels) formalize this and are useful as a shared internal vocabulary.

- The right autonomy level for a given action depends on reversibility and stakes, not on model capability: a highly capable model doing something irreversible and high-stakes still warrants a checkpoint; a low-stakes reversible action doesn't need one even from a less capable model.
- Approval UX has to show the *plan*, not just ask for a yes/no: a confirmation dialog that doesn't let the user see what specifically will happen (which records, which recipients, which amount) is a rubber stamp, not oversight.
- Default friction should be intentional and preserved in some contexts (compliance-sensitive actions, financial transactions) even as autonomy elsewhere increases: removing friction indiscriminately in the name of "frictionless AI" recreates the exact failure mode this principle exists to prevent.
- Autonomy level should be an explicit, changeable configuration tied to demonstrated reliability in your own environment, not a fixed vendor default accepted at install time.

Questions I Should Be Able to Ask My Team:

1. For each AI agent with write access to production systems, what autonomy level does it operate at, and what evidence justified that level rather than a more conservative one?
2. When our agents ask for human approval, does the interface show the actual plan and its consequences, or just a generic confirm/cancel button?
3. What is our process for increasing an agent's autonomy level over time, and what would trigger us to reduce it after an incident?

Technologies / Standards / Companies to Know: Human-in-the-loop / human-on-the-loop patterns, agent action logging and audit trails, "Levels of Autonomy for AI Agents" taxonomy

Recommended Learning:

- Levels of Autonomy for AI Agents: Knight First Amendment Institute: a rigorous, non-vendor framework for structuring autonomy levels, useful as internal shared language.
- AI Agents as Users: Nielsen Norman Group: see Topic 6; also covers where friction should deliberately remain for agent actions.

Time Investment: 2-3 hours

18.3 Citations, Provenance, Confidence Indicators & AI Failure UX

Priority: Monitor

Executive Definition: This is the design discipline of making an AI system's uncertainty and sourcing visible and actionable: showing where an answer came from (provenance), how confident the system actually is (calibrated confidence, not decorative), and what to do when it's wrong (failure UX): rather than presenting every output with the same uniform, polished confidence regardless of accuracy. Nielsen Norman Group research documents that current chatbot interface conventions actively work against this, making outputs look more trustworthy than they are.

Why It Matters: Published research finds generative AI hallucination rates in the range of 13.5%–33% depending on task, yet interface conventions (confident tone, polished formatting, warnings that scroll off screen) create a "halo effect" that discourages users from verifying outputs: meaning the UX itself, not just model accuracy, is a source of real business risk when unverified AI output reaches customers, financial records, or compliance-sensitive decisions.

What I Need to Understand:

- NN/g research shows chatbot interfaces are currently designed in ways that *discourage* error-checking (authoritative tone, formatting polish, disclaimers that disappear from view): this is a design failure, not solely a model-accuracy failure, and is fixable independent of model improvements.
- The rule of thumb from NN/g's research on trust: users should only rely on AI output they can verify or already know to be true, and should stay within their own domain of expertise when accepting AI claims at face value: a principle that should show up in your internal AI usage guidance, not just in UX design.
- Verification should be designed to be as low-effort as generation: inline source links, clickable claims, and prompts that invite scrutiny ("what's your confidence here," "what would change this answer") outperform generic disclaimers.
- Confidence indicators are only useful if calibrated to actual accuracy; a system that displays uniform confidence regardless of correctness is worse than no indicator, because it actively misleads.
- Failure UX (what happens when the AI is wrong, unavailable, or refuses) deserves as much design investment as the success path: most AI product reviews focus entirely on the happy path.

Questions I Should Be Able to Ask My Team:

1. In our customer- or employee-facing AI features, is there any confidence signal at all, and if so, has anyone validated that it's calibrated to actual accuracy rather than decorative?
2. Do our interfaces make it easy to trace an AI claim back to its source, or does the user have to take the output on faith?
3. What does our product do when the AI is wrong, uncertain, or can't complete a request: has that path been designed deliberately, or does it default to a generic error?

Technologies / Standards / Companies to Know: RAG (retrieval-augmented generation) citation patterns, confidence calibration techniques, NN/g chatbot design guidelines

Recommended Learning:

- When Should We Trust AI? Magic-8-Ball Thinking and AI Hallucinations: Nielsen Norman Group: hallucination rate data and a practical decision framework for when AI output is safe to accept.
- AI Chatbots Discourage Error Checking: Nielsen Norman Group: the core research on why current chatbot UX suppresses verification behavior.

- Explainable AI in Chat Interfaces: Nielsen Norman Group: design patterns for making AI reasoning and sourcing legible.

Time Investment: 1 hour

Organizational Adoption & Change Management

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/organizational-adoption-and-change-management/

19.1 Executive & Role-Based AI Literacy

Priority: Must Understand

Executive Definition: Role-based AI literacy means the level and kind of AI understanding required differs by job: a developer needs to understand agentic tool-use and code review implications, a knowledge worker needs to understand appropriate use and data handling, and an executive needs enough understanding to make investment, risk, and governance calls without being talked past by vendors or by their own technical staff. This is now a compliance requirement in some jurisdictions, not just good practice.

Why It Matters: The EU AI Act's Article 4 (in force since February 2025) legally requires providers and deployers of AI systems to ensure staff "involved in the operation and use of AI systems possess a sufficient degree of AI literacy," proportionate to each person's role, technical background, and the system they use: a binding obligation for any organization operating in or serving the EU, not a training nice-to-have. Separately, Gallup found only 8% of employees strongly agree that "AI has transformed how work gets done" at their organization despite 65% of AI users reporting improved personal productivity: a gap that typically signals literacy and process redesign lagging tool rollout, which is exactly what a literacy program is meant to close.

What I Need to Understand:

- Article 4's proportionality principle: literacy requirements scale with role and risk, not a flat one-size-fits-all training module: your program needs role-based tiers (executive, developer, general knowledge worker) not one deck for everyone
- The difference between tool literacy (how to use a specific product) and risk literacy (what can go wrong, when to escalate, what not to trust): most vendor-provided training covers only the former
- Your own personal obligation as an executive setting AI strategy: enough literacy to challenge a vendor's claims and your own technical team's assumptions, not enough to build a model yourself
- What "sufficient degree" means in practice for compliance purposes: there's no certification standard yet, so documentation of your training program and its rationale is currently your best evidence of compliance
- The gap between adoption and transformation Gallup identifies: literacy alone doesn't close it, but its absence guarantees it stays open

Questions I Should Be Able to Ask My Team:

1. Do we have role-differentiated AI literacy content (executive, developer, general staff), or is everyone getting the same generic training regardless of what they actually do with AI?

2. Can we produce documentation showing who has completed what AI literacy training, mapped to their role and the systems they use: the evidence an EU regulator or auditor would ask for?
3. What's our plan for keeping literacy content current as the tools and risks change: is this a one-time rollout or a maintained program?

Technologies / Standards / Companies to Know: EU AI Act Article 4, NIST AI RMF (as a literacy content source), ISO/IEC 42001.

Recommended Learning:

- Article 4: AI literacy: EU Artificial Intelligence Act: plain-language explainer of the legal text, with links to the official regulation.
- Regulation (EU) 2024/1689 (EUR-Lex, official text): the actual binding legal text, Article 4 specifically, for anyone who needs to cite the obligation precisely.
- AI Risk Management Framework (NIST): a solid content source for the "risk literacy" half of a program, independent of the EU legal requirement.

Time Investment: 1 hour

19.2 AI Champions, Communities of Practice & Use-Case Libraries

Priority: Should Understand

Executive Definition: An AI champions program identifies motivated employees embedded in business teams (not the central AI/platform team) who model real usage, mentor peers, and surface what's actually working, functioning as a peer-driven adoption layer that a top-down mandate or training rollout can't replicate. A use-case library is the artifact that program should produce: a growing, curated internal record of validated AI applications, so teams stop reinventing the same pilot in five different departments.

Why It Matters: GitHub's own internal playbook for this (a company whose product is developer tooling, so its adoption playbook is directly relevant to your developer population) treats champions as essential specifically because "buying AI tools without empowering people to use them is a fast track to failure." For an organization with both a large engineering population and a large professional/knowledge-worker population, a single central team cannot realistically drive grounded adoption in both; champions are the mechanism that scales adoption without scaling the central team headcount 1:1 with the workforce.

What I Need to Understand:

- Champions are peer-selected/volunteer, not appointed by management: mandating "champions" top-down tends to produce compliance theater rather than genuine peer influence
- The three-phase pattern GitHub documents: recruit volunteers (~30 days), build community infrastructure and cadence (~90 days), then hand ownership to the community itself rather than keeping it centrally run indefinitely

- A use-case library only has value if it's curated (validated, with outcomes attached) rather than an unfiltered wiki of every experiment anyone tried: curation is the actual work, not the tooling
- How this connects to your operating model (see AI Operating Models topic): champions are the informal complement to a formal hub-and-spoke structure, covering the reach a central CoE can't
- Measurement is qualitative-plus-quantitative by design (adoption stories plus usage metrics): don't expect or demand a single clean ROI number from a champions program

Questions I Should Be Able to Ask My Team:

1. Are our AI champions volunteers with real peer credibility on their teams, or are they people we assigned the title because someone needed to own it?
2. What does our use-case library actually contain: validated, outcome-tagged examples, or an unfiltered list of everything anyone has tried?
3. If we stopped centrally running the champions program today, would it survive on its own, or does it collapse the moment central attention moves elsewhere?

Technologies / Standards / Companies to Know: GitHub (publishes the most concrete public playbook), OpenAI Academy champion role guidance: read these as case studies from AI-tooling vendors, with the obvious caveat that they have an interest in you adopting more AI tools.

Recommended Learning:

- Activating your internal AI champions (GitHub): the most concrete, structured public playbook, including the three-phase rollout timeline.
- The AI Champion role (OpenAI Academy): a second vendor's framing of the same role, useful for comparing what's common across both versus specific to one company's tools.

Time Investment: 30 minutes

19.3 Employee Displacement Fear, Change Management & Acceptable Use Policies

Priority: Must Understand

Executive Definition: This topic covers three linked things you're accountable for as a leader: managing genuine, measurable employee anxiety about AI-driven job loss; running change management that acknowledges that anxiety instead of talking past it; and publishing an Acceptable Use Policy (AUP) that tells employees clearly what AI tools they may use, on what data, and under what constraints: closing the gap that otherwise gets filled by unsanctioned "shadow AI" use.

Why It Matters: Gallup found 18% of all U.S. employees believe it's likely their job will be eliminated by AI within five years, rising to 23% among employees at organizations that have adopted AI, and that fear is not evenly distributed, meaning your own workforce's number could be materially higher depending on role exposure. Organizations that have adopted AI also report both more hiring (34% vs. 28%) and

more workforce reduction (23% vs. 16%) than non-adopters, so "AI means fewer jobs" is not simply true or false at the org level: it means more workforce churn in both directions, which is itself what needs managing, not just the layoff fear.

What I Need to Understand:

- Gallup's finding that fear is rising with adoption (15% in mid-2024 to 23% now among AI adopters): meaning your own rollout, if unmanaged, likely increases anxiety even as it increases productivity, and treating these as automatically offsetting is a mistake
- The distinction between managing fear (communication, transparency, career-pathing) and managing actual displacement (real headcount decisions): conflating the two either dismisses legitimate fear or over-promises job security you can't guarantee
- What a real Acceptable Use Policy must specify concretely: which tools are sanctioned, what data classifications may/may not be sent to which tools, human-review requirements for AI output in specific workflows, and consequences for violation: a vague "use AI responsibly" memo is not a policy
- The OWASP LLM Top 10's "Excessive Agency" and "Sensitive Information Disclosure" categories are the concrete risks a good AUP is written to prevent, giving you a technical anchor for policy language rather than only an HR one
- Only 8% of employees strongly agree AI has "transformed how work gets done" at their organization (Gallup) even where usage is high: meaning most of your workforce likely still experiences AI as an add-on tool, not a redesigned way of working, which is itself a change-management gap

Questions I Should Be Able to Ask My Team:

1. Have we actually measured employee sentiment and job-security fear related to our AI rollout, or are we assuming it's fine because usage numbers are up?
2. Can I see our current Acceptable Use Policy, and does it name specific data classifications and specific approved tools, or is it a page of generic principles?
3. What happens today, concretely, to an employee who pastes customer PII into an unsanctioned AI tool: is there a real enforcement mechanism, or does the policy exist only on paper?

Technologies / Standards / Companies to Know: NIST AI RMF (Govern function: the basis for AUP structure), OWASP Top 10 for LLM Applications (technical risk categories an AUP should reflect), Gallup workplace research (ongoing employee sentiment tracking).

Recommended Learning:

- Rising AI Adoption Spurs Workforce Changes (Gallup): primary survey data on displacement fear, hiring/reduction patterns, and the productivity-sentiment gap.
- AI Risk Management Framework (NIST): the Govern function is the right structural reference for building an AUP that's more than a memo.

- OWASP Top 10 for LLM Applications 2025: concrete risk categories (sensitive information disclosure, excessive agency) an AUP needs to actually address.

Time Investment: 2-3 hours

AI Measurement & Business Value

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/ai-measurement-and-business-value/

20.1 Engineering Metrics in the AI Era (DORA Metrics + AI-Specific Measures)

Priority: Must Understand

Executive Definition: DORA's four/five keys: deployment frequency, lead time for changes, change failure rate, time to restore service, and (in more recent research) measures of rework/churn: remain the standard for software delivery performance, but DORA's own current research is explicit that AI coding assistants are changing what these metrics mean: code gets generated faster, but verification, review, and integration overhead grow, so throughput metrics alone can improve while real delivery quality does not. DORA's research specifically warns against "tokenmaxxing" (treating AI token/prompt volume as a performance indicator) and recommends selecting from DORA, SPACE, or DevEx frameworks based on what your organization actually needs to learn, not defaulting to whichever is easiest to instrument.

Why It Matters: Engineering leaders under pressure to show "AI ROI" are the group most likely to reach for the easiest available number (lines of code generated, percentage of code "AI-assisted," tool licenses deployed) and DORA's research is a direct, credible warning that these are exactly the wrong signals, because AI can inflate them while quietly increasing verification burden, skill degradation, and change failure rate elsewhere in the pipeline.

What I Need to Understand:

- DORA's core metrics (deployment frequency, lead time, change failure rate, time to restore) still matter, but interpret them alongside stability metrics specifically: AI acceleration of the "throughput" side without matching improvement in "stability" metrics is a warning sign, not a win.
- DORA research documents a real paradox: developers using AI tools report higher satisfaction while also reporting they spend less time on work they consider valuable: satisfaction and productivity are not the same signal, and a rise in one doesn't validate the other.
- "Tokenmaxxing" (rewarding or measuring AI adoption by volume of tokens/prompts consumed) is explicitly flagged by DORA as a dangerous trap, because it incentivizes usage rather than outcomes (this is the engineering-specific instance of the broader Topic 12 problem).
- AI coding assistants introduce hidden costs not captured by classic delivery metrics: verification overhead, skill degradation risk for less experienced engineers, and integration friction: these need their own tracking, not just an assumption that faster code generation equals faster delivery.
- No single framework (DORA vs. SPACE vs. DevEx) is sufficient alone; DORA's own guidance is to choose the framework(s) that match what you're actually trying to learn about your organization, and to expect that to include qualitative developer-experience signals alongside quantitative

delivery metrics.

Questions I Should Be Able to Ask My Team:

1. Since introducing AI coding tools, have our change failure rate and time to restore moved in the same direction as our deployment frequency and lead time, or is throughput up while stability is flat or worse?
2. Are we measuring or incentivizing anything by raw AI usage volume (tokens, prompts, "percent AI-assisted code"), and if so, can we defend that against DORA's tokenmaxxing warning?
3. Do we have any signal on verification overhead or code churn/rework tied to AI-generated code, or are we only looking at generation speed?

Technologies / Standards / Companies to Know: DORA (DevOps Research and Assessment), SPACE framework, DevEx framework, AI coding assistants (GitHub Copilot, Claude Code, Cursor, etc.: evaluate neutrally per vendor)

Recommended Learning:

- DORA Insights: the primary source for current DORA research and reports.
- DORA research on AI: DORA's dedicated body of work on AI's effect on software delivery, including the ROI of AI-assisted development report.

Time Investment: 2-3 hours

20.2 Knowledge-Worker Productivity Metrics

Priority: Must Understand

Executive Definition: Measuring AI's effect on non-engineering knowledge work requires task-level output metrics (units completed, quality, time-to-resolution) tied to a credible baseline and, critically, segmented by worker skill/experience level: because the best available field research shows AI's productivity effect is highly uneven across a workforce, not a uniform lift you can apply as a flat multiplier.

Why It Matters: The most rigorous field study to date (a large-scale study of customer support agents using a generative AI assistant) found a 14% average productivity gain but with the gain concentrated almost entirely among novice and lower-skilled workers (~34% gain) while experienced workers saw little to no measurable benefit: an average number here actively hides the story, and any executive reporting a single average productivity figure across a whole knowledge-worker population is very likely obscuring who is actually benefiting and why.

What I Need to Understand:

- Aggregate/average productivity figures across a knowledge-worker population are close to meaningless without segmentation by experience or skill level: the underlying research shows the effect is concentrated, not uniform.

- The strongest documented mechanism for AI productivity gains in knowledge work is diffusion of best practices from top performers to less experienced workers, effectively compressing the learning curve: this is a different and more measurable claim than generic "AI makes people more productive."
- Task-level output metrics (issues resolved per hour, time-to-resolution, quality/error rate, customer satisfaction) are more defensible than time-saved self-reports, which are notoriously unreliable and inflated.
- Retention and skill-development effects (does AI use correlate with faster ramp-up for new hires, lower attrition) are real, measurable secondary benefits documented in the research and worth tracking alongside raw output.
- Any productivity claim needs a comparison group or a credible pre/post baseline: anecdote-driven claims ("teams say they're faster") are not evidence and should be treated skeptically at the executive level.

Questions I Should Be Able to Ask My Team:

1. When we report a knowledge-worker productivity gain from AI, is it an average across everyone, or do we know which segments (tenure, skill level) are actually driving it?
2. Do we have task-level output metrics with a real baseline for any AI-assisted knowledge work function, or are we relying on self-reported time savings?
3. Are we seeing evidence that AI use is compressing ramp-up time for new hires or narrowing the gap between our best and average performers: the effect the strongest research predicts?

Technologies / Standards / Companies to Know: Task-level output/quality metrics, controlled rollout / A-B methodology for internal AI tools, National Bureau of Economic Research (NBER) field-experiment methodology

Recommended Learning:

- Generative AI at Work: Brynjolfsson, Li, Raymond (NBER Working Paper 31161): the primary field-experiment source for the skill-segmented productivity finding; read this before accepting any vendor productivity claim.

Time Investment: 1 hour

20.3 Enterprise Value Metrics (Revenue Enabled, Cost Reduction, Capacity Released, Adoption/Workflow Penetration)

Priority: Must Understand

Executive Definition: Enterprise AI value should be tracked through four outcome-linked categories: revenue enabled (new revenue or deals directly attributable to AI-enabled capability), cost reduction (measured, realized cost removed from a process, not projected savings), capacity released (hours or headcount-equivalent freed up and (critically) what was actually done with that freed capacity), and

adoption/workflow penetration (the percentage of a given workflow's volume actually running through the AI path, not the percentage of people with a license). These are outcome metrics; everything in Topic 12 is what they are not.

Why It Matters: Most enterprise AI reporting collapses into activity metrics (usage, engagement, satisfaction scores) because they're easy to instrument, but none of them answer the question a board or CFO actually asks: did this change the economics of the business. Capacity released is the category most often mismeasured or left untracked entirely: freeing up hours that are never redeployed to higher-value work is not value capture, it's a cost center running under capacity, and reporting the hours freed without reporting what happened to them is close to fabrication.

What I Need to Understand:

- Revenue enabled needs a defensible attribution chain (which deals, which capability, what would have happened otherwise): "AI-related revenue" claims without that chain are marketing, not measurement.
- Cost reduction should be realized (removed from the budget) not merely modeled/projected: a business case showing hypothetical savings is not the same claim as a finance-verified reduction.
- Capacity released is only a real value metric when paired with a redeployment answer: what is now being done with the freed time/headcount: absorbed into other work, redeployed to new initiatives, or actually reduced from the cost base. Track both halves or the metric is incomplete.
- Adoption/workflow penetration should be measured as share of eligible task/transaction volume actually flowing through the AI-enabled path, not share of licensed seats or logged-in users (see Topic 12): penetration of the workflow, not penetration of the roster.
- These four categories should roll up to a small, consistent executive dashboard reviewed on a cadence (quarterly is reasonable) rather than a sprawling set of team-specific metrics that can't be compared or aggregated.

Questions I Should Be Able to Ask My Team:

1. For every capacity-released claim we report, can we show what the freed time/headcount was actually redeployed to, or are we just reporting hours saved with no destination?
2. What percentage of the *eligible transaction volume* for this workflow is actually running through the AI-enabled path, as opposed to what percentage of licensed users have logged in?
3. Walk me through the attribution chain for our largest AI-attributed revenue or cost-reduction claim: what would we expect to see if it were true, and have we checked?

Technologies / Standards / Companies to Know: Workflow-level instrumentation/analytics, finance-verified savings tracking (vs. modeled), DORA's "ROI of AI-Assisted Software Development" methodology as a template for rigor

Recommended Learning:

- DORA research on AI: see Topic 9; the ROI framework there is a useful template for value-metric rigor even outside engineering.

- Generative AI at Work: NBER Working Paper 31161: see Topic 10; a rare example of a redeployment/value claim backed by controlled measurement rather than survey data.

Time Investment: 2-3 hours

20.4 Why Prompt Counts & Seat/User Counts Are Vanity Metrics

Priority: Should Understand

Executive Definition: Prompt volume, login counts, and licensed-seat totals measure whether people touched a tool, not whether anything about the business changed as a result. They are the AI-era equivalent of "page views": trivially gameable, easy to report, and structurally incapable of answering whether a workflow got faster, cheaper, better, or eliminated. Any AI reporting that leads with these numbers should be treated as unverified until translated into the outcome metrics in Topic 11.

Why It Matters: These are the numbers every AI vendor and most internal teams default to reporting because they require no attribution work and always trend upward: which is precisely why they're worthless as evidence of value and dangerous as a basis for investment decisions.

What I Need to Understand:

- A seat count measures procurement, not transformation. A prompt count measures curiosity, not capability change. Neither tells you whether a single workflow got faster, cheaper, or better: that is the sentence to hold every AI status report against.
- DORA's own research names the engineering-specific version of this problem directly: "tokenmaxxing" (treating AI token/prompt spend as a performance indicator) as a dangerous trap that rewards volume over outcome (see Topic 9); the same logic applies to any usage count, in any function.
- Usage metrics are trivially inflatable without any underlying behavior change: mandate logins, auto-generate prompts, count every keystroke as "engagement": none of it requires the workflow to actually improve.
- The NBER field study on AI productivity (Topic 10) shows *access* to a tool and *realized productivity gain* are entirely different things that vary enormously by user: a company reporting "80% of employees are active AI users" has told you nothing about whether those users are the 34%-gain novices or the near-zero-gain experts, or anything in between.
- The correct question is never "how much was it used": it's "which workflow changed, by how much, and what's the evidence." If a metric can go up while zero business outcomes change, it is a vanity metric, full stop.

Questions I Should Be Able to Ask My Team:

1. When someone reports "X% of employees are active AI users" or "Y prompts sent last month," what workflow-level outcome are they actually claiming, and can they show it?

2. If we doubled our prompt volume or seat count next quarter with zero change to any workflow's speed, cost, or quality, would our current reporting even notice? If not, our reporting is broken.
3. For every usage metric in our current AI dashboard, is there a corresponding outcome metric (from Topic 11) it's supposed to be a proxy for, and does the proxy actually correlate with the outcome in our own data?

Technologies / Standards / Companies to Know: DORA's "tokenmaxxing" critique, outcome-based analytics vs. engagement analytics, workflow-level instrumentation

Recommended Learning:

- DORA research on AI: see Topic 9; the primary credible source explicitly naming usage-volume-as-performance as a trap.
- Generative AI at Work: NBER Working Paper 31161: see Topic 10; direct evidence that tool access and realized gain diverge sharply by user segment, undercutting any flat usage metric.

Time Investment: 30 minutes

Emerging Technology Radar

Last updated September 11, 2026 · 13 min read · https://erikcaldwell.com/field-guide/emerging-technology-radar/

For each item below: classification (Near-term / Developing / Highly Uncertain-Speculative), what it is, why it matters enough to watch, and the concrete signal that would indicate it's moving from speculative toward real.

21.1 Autonomous Software Development (issue-to-PR-to-deploy)

Classification: Developing **What it is:** Coding agents (Devin, GitHub Copilot coding agent, Cursor/Cognition background agents, Claude Code, etc.) that take a ticket, write and test code, open a PR, and in some pipelines merge and deploy with little or no human step in between. **Why watch it:** If reliable, this changes engineering headcount math and where review/QA effort goes, but reliability, not model capability, is the binding constraint at present. **Signal to track:** SWE-bench Verified scores and, more importantly, real production incident/rollback rates disclosed by early adopters running agents with merge authority (not sandboxed benchmarks).

21.2 Self-Healing Software

Classification: Developing **What it is:** AI systems that detect a production fault, diagnose root cause, and remediate (rollback, restart, patch, reconfigure) without a human in the loop. **Why watch it:** Narrow, bounded self-healing (auto-rollback, known-failure runbooks) is already viable and cuts MTTR; general-purpose autonomous remediation of novel failures is not: the gap between vendor claims and audited outcomes is wide. **Signal to track:** Whether a major APM/AIOps vendor (Datadog, PagerDuty, Dynatrace) publishes audited MTTR/false-remediation data for closed-loop (no human approval) fixes, not just "AI-assisted" ones.

21.3 Software Factories (fleets of coding agents as a production system)

Classification: Developing **What it is:** Treating dozens-to-hundreds of coding agents as a managed production system (with queues, review gates, evals, and observability) rather than one agent per developer. **Why watch it:** This is the operating-model question behind autonomous coding: it determines whether agent output scales safely or just scales incident volume. **Signal to track:** Vendors like Factory.ai publishing fleet-scale reliability/throughput metrics, and whether a large enterprise (not a startup) discloses agent-fleet headcount-equivalent numbers.

21.4 Agent Marketplaces

Classification: Developing **What it is:** Storefronts for pre-built third-party agents that plug into an enterprise's stack: Salesforce's AgentExchange (successor to AppExchange) is the clearest concrete example, alongside Microsoft's and Google's agent catalogs. **Why watch it:** Marketplaces shift agent risk from "we built it" to "we procured it," which raises new vendor-vetting, security, and liability questions procurement isn't yet set up for. **Signal to track:** Whether AgentExchange-style marketplaces publish real usage/trust metrics (installs, incident disclosures) rather than just listing counts. (salesforce.com)

21.5 Agent Discovery Protocols

Classification: Near-term **What it is:** Standards (Anthropic's Model Context Protocol for tool/data access; Google's Agent2Agent/A2A for agent-to-agent interop, now under the Linux Foundation) that let agents find and call each other's capabilities. **Why watch it:** MCP has become the de facto integration layer for agent tooling in about two years: protocol consolidation (or fragmentation) directly affects integration cost and vendor lock-in risk. **Signal to track:** The Linux Foundation reports A2A passed 150+ member organizations with early enterprise production use; watch whether MCP and A2A converge, merge governance, or start competing for the same layer. (linuxfoundation.org, developers.googleblog.com)

21.6 Agent Reputation Systems

Classification: Highly Uncertain/Speculative **What it is:** Mechanisms to score an agent's trustworthiness/track record before letting it transact or act on your behalf: proposals range from centralized marketplace ratings to on-chain registries (e.g., Ethereum's ERC-8004 "trustless agents" proposal). **Why watch it:** Without reputation, agent-to-agent commerce and open marketplaces can't scale past known counterparties: it's a prerequisite, not a nice-to-have, for items 4, 7, 27 and 28. **Signal to track:** Any agent marketplace (Salesforce, Google, OpenAI) shipping a real reputation/score field that affects transaction routing, versus reputation remaining a marketing claim.

21.7 Agent Payments / Agentic Commerce

Classification: Developing **What it is:** Protocols letting an AI agent initiate and complete a purchase on a human's or business's behalf: Google's Agent Payments Protocol (AP2, with Mastercard, Coinbase, and others), and the OpenAI/Stripe Agentic Commerce Protocol behind ChatGPT's Instant Checkout. **Why watch it:** Real card networks and payment processors are already building rails for this; enterprise exposure (procurement agents making purchases) will arrive faster than most CFOs expect. **Signal to track:** Whether AP2/ACP transaction volume moves beyond consumer shopping demos into Business-to-Business (B2B) procurement, and whether card networks publish agent-initiated fraud/chargeback rates. (cloud.google.com, stripe.com)

21.8 Agent Identity Standards (beyond OAuth)

Classification: Developing **What it is:** Purpose-built identity systems for non-human actors: Microsoft's Entra Agent ID (reached GA in 2026) is the clearest example, giving each agent its own directory identity, lifecycle, and permissions distinct from a service account or a human's delegated OAuth token. **Why watch it:** Today's OAuth-based patterns weren't designed for entities that spawn sub-agents and act continuously; identity sprawl is already a named problem in early Entra Agent ID deployments. **Signal to track:** Cross-vendor interop: whether an Entra Agent ID can be recognized/verified by a non-Microsoft agent platform, not just within Microsoft's stack. (learn.microsoft.com)

21.9 Persistent Digital Workers (named, long-lived agent "employees")

Classification: Developing **What it is:** Agents given a persistent identity, role, and memory across sessions (framed explicitly as "digital labor" by Salesforce and similarly by Microsoft's Copilot agent framing) rather than stateless per-task tools. **Why watch it:** This is the framing vendors are using to sell per-seat/per-outcome pricing for agents; the actual persistence and memory quality (versus a rebranded chatbot) determines whether it's substance or packaging. **Signal to track:** Independent (non-vendor) case studies quantifying a named digital worker's task success rate and cost-per-outcome over months, not launch-day demos. (salesforce.com)

21.10 Autonomous Departments

Classification: Highly Uncertain/Speculative **What it is:** An entire business function (e.g., collections, tier-1 support, basic procurement) run predominantly by coordinated agents with a human only in an oversight role. **Why watch it:** This is the aggregation point where individual agent ROI either compounds into department-level headcount change or stays a productivity veneer: worth tracking because it's the scenario most consultancies (Deloitte, BCG) are modeling toward 2027-2028, not one broadly achieved today. **Signal to track:** A named, audited case of a function running with a materially reduced human headcount for over a year (not a pilot announcement). (deloitte.com)

21.11 Synthetic Organizations

Classification: Highly Uncertain/Speculative **What it is:** The theoretical end state of the above: an organization whose operating structure is substantially composed of interacting agents rather than human teams, sometimes discussed in academic/futurist contexts as multi-agent "societies." **Why watch it:** Worth naming precisely so it isn't confused with today's real (and much more limited) digital-worker and autonomous-department efforts: the absence of credible reporting on this happening anywhere in 2026 is itself the calibrating signal. **Signal to track:** Any peer-reviewed or analyst-verified example, as distinct from vendor thought-leadership content: none currently meets that bar.

21.12 AI-Native Enterprise Resource Planning (ERP)

Classification: Developing **What it is:** ERP vendors (SAP with Joule agents, Oracle Fusion, Microsoft Dynamics) adding agentic layers on top of existing systems of record, short of a ground-up re-architecture around agents. **Why watch it:** Most "AI-native ERP" today is agent bolt-ons to legacy data models: the strategic question is whether incumbents or challengers get to the re-architecture first, since that's where lock-in shifts. **Signal to track:** Whether SAP or Oracle ships agent-initiated transactions (e.g., autonomous PO creation/approval) as a default workflow rather than an assisted suggestion, with disclosed error rates.

21.13 AI-Native CRM

Classification: Developing **What it is:** Salesforce Agentforce and HubSpot Breeze are the two clearest branded pushes toward CRM where agents (not just automations) handle lead qualification, service cases, and follow-up autonomously. **Why watch it:** CRM is the highest-volume, lowest-stakes-per-transaction place to pressure-test agent autonomy in a live revenue system before extending it to higher-stakes domains. **Signal to track:** Independent (Gartner/Forrester) satisfaction and containment-rate data for Agentforce/Breeze deployments, versus vendor-published case studies.

21.14 Generative UI (interfaces generated on the fly)

Classification: Developing **What it is:** Interfaces assembled dynamically per query/task by a model rather than pulled from a fixed set of pre-built screens: Google Research has published this explicitly as a direction ("a rich, custom, visual interactive experience for any prompt"). **Why watch it:** If it works, it collapses the cost of building bespoke internal tools/dashboards; if it doesn't, it produces inconsistent, unreviewable UI sprawl: the design and accessibility tooling to govern this at scale doesn't exist yet. **Signal to track:** Whether a major consumer product (Search, Gemini app) ships generative UI as a default experience rather than an experimental toggle. (research.google)

21.15 Software Generated on Demand (ephemeral, single-use apps)

Classification: Developing **What it is:** Disposable, single-purpose applications a model generates for one task and discards: already visible in miniature via features like Claude's Artifacts or ChatGPT's Canvas generating a one-off tool mid-conversation. **Why watch it:** This challenges the economics of building and maintaining internal software at all for narrow, recurring-but-low-volume needs, but raises real questions about security review, data governance, and audit trail for code no one ever formally ships. **Signal to track:** Whether any enterprise formally sanctions (with a governance policy) employee-generated ephemeral apps touching real business data, versus treating all such use as shadow IT.

21.16 Natural-Language Programming (non-developers directing agents)

Classification: Developing **What it is:** "Vibe coding" (a term popularized by Andrej Karpathy in early 2025) where someone without formal engineering training describes intent in natural language and an agent (Replit, Lovable, Cursor, Claude Code, base44, etc.) produces working software. **Why watch it:** This is the most direct threat/opportunity to the "who is allowed to build software" boundary in a large engineering org: shadow engineering by knowledge workers is already happening, whether or not IT has a policy for it. **Signal to track:** Whether your own knowledge workers are already producing production-adjacent tools this way (check for unsanctioned Replit/Lovable/Claude-built internal tools) before deciding this is someone else's problem.

21.17 Edge AI

Classification: Near-term **What it is:** Running inference on devices/gateways at the network edge rather than in the cloud, for latency, cost, connectivity, or data-residency reasons. **Why watch it:** This is a mature, well-funded infrastructure category already embedded in manufacturing, retail, and industrial IoT deployments: the executive question is procurement/architecture, not "should we watch this." **Signal to track:** Total cost of ownership crossover points as edge hardware (NPUs) gets cheaper: track your own inference cost mix moving from cloud API calls to edge/on-prem for high-volume, low-complexity tasks.

21.18 Local / On-Device Models

Classification: Near-term **What it is:** Capable small models running entirely on a phone, laptop, or PC without a network call: Apple Intelligence (reportedly now built in part on licensed Google Gemini models per 2026 reporting), Gemini Nano, Microsoft's Phi family, and Llama variants are the concrete examples. **Why watch it:** This determines the floor for what "AI" means when connectivity, latency, or data-privacy requirements rule out a cloud call: increasingly relevant for regulated data and field/offline use cases. **Signal to track:** Whether your device fleet (laptops issued to your knowledge workers) ships with a capable local model by default, changing what needs cloud API budget at all. (appleinsider.com)

21.19 Embodied AI

Classification: Highly Uncertain/Speculative **What it is:** AI systems that perceive and act in the physical world through a body (robot, drone, vehicle) rather than through text/screen interfaces, using the same foundation-model techniques as LLMs. **Why watch it:** Almost entirely irrelevant to a software/knowledge-work enterprise directly, but relevant as a leading indicator of foundation-model capability generalizing beyond language, and China's state-level push (per Merics reporting) is a geopolitical signal worth tracking even if you never buy a robot. **Signal to track:** Independent, non-demo-reel evidence of embodied agents completing multi-step real-world tasks reliably outside controlled lab/showroom settings. (merics.org)

21.20 Robotics (AI-driven, foundation-model-based)

Classification: Developing **What it is:** Vision-language-action (VLA) foundation models controlling physical robots: Figure AI's Helix model and Physical Intelligence's π-0 are the clearest named examples, alongside Chinese firms like AgiBot. **Why watch it:** Same reasoning as embodied AI generally, but this is the layer with actual funded companies, named models, and pilot deployments (logistics, manufacturing) rather than pure research: worth distinguishing from the broader speculative category above. **Signal to track:** A named humanoid/robot deployment moving from a single-site pilot to multi-site commercial contract with disclosed unit economics. (en.wikipedia.org)

21.21 World Models

Classification: Highly Uncertain/Speculative **What it is:** Models trained to build an internal predictive simulation of how the physical/visual world evolves (rather than just predicting text), exemplified by DeepMind's Genie line and Fei-Fei Li's World Labs (Marble). **Why watch it:** Proponents argue world models are a precondition for robust embodied AI and long-horizon planning; skeptics note none has yet demonstrated enterprise utility beyond generating interactive video/game environments. **Signal to track:** Any world model demonstrating a real economic use case beyond content/simulation generation (e.g., materially improving robotic task success or supply-chain simulation) versus remaining a research demo.

21.22 Digital Twins (AI-enhanced)

Classification: Developing **What it is:** Physics- or data-based simulations of a real asset, process, or system (a factory line, a supply chain, a building), now increasingly layered with AI for prediction and what-if reasoning: Nvidia Omniverse and Siemens Xcelerator are the established commercial platforms. **Why watch it:** Unlike most items on this list, digital twins are already operationally mature in manufacturing/industrial contexts: the "watch" item is specifically the AI layer (generative simulation, agent-driven scenario testing) being added on top of an already-proven category. **Signal to track:** Whether AI-driven scenario generation in a digital twin platform demonstrably shortens a real capital-planning or process-redesign cycle, versus remaining a visualization upgrade.

21.23 Synthetic Data (for training/eval at enterprise scale)

Classification: Near-term **What it is:** Artificially generated data used to train or evaluate models when real data is scarce, sensitive, or imbalanced: already central to frontier model training (Nvidia's Nemotron pipelines, Microsoft's Phi models trained substantially on synthetic data). **Why watch it:** For a large enterprise engineering organization, synthetic data is the practical near-term path to fine-tuning/evaluating internal models on sensitive data (code, customer records) without exposing the real thing: governance and quality-control practices matter more than the underlying tech. **Signal to track:** Whether your own model-eval or fine-tuning pipelines have an explicit synthetic-data quality/bias-audit step, not just a generation step.

21.24 AI for Scientific Discovery

Classification: Developing **What it is:** AI systems directly accelerating scientific research: DeepMind's AlphaFold (2024 Nobel Prize in Chemistry to Hassabis and Jumper), GNoME for materials discovery, and Google's "AI co-scientist" system (announced 2025) are the clearest proof points. **Why watch it:** These are the most credible, least-hyped evidence that foundation models generalize beyond language to genuine scientific value: relevant to any enterprise with an R&D, pharma, materials, or advanced-engineering function. **Signal to track:** Whether AI-assisted discoveries (new materials, drug candidates) move from computational prediction into validated real-world results at a rate distinguishable from normal R&D baseline.

21.25 Real-Time Multimodal Agents (voice + vision + action, low latency)

Classification: Developing **What it is:** Agents that see, hear, and respond in real time with low enough latency for natural conversation and action: OpenAI's GPT-4o Realtime API and Google's Project Astra/Gemini Live are the current reference implementations. **Why watch it:** This is the interface layer that could replace app/dashboard-based interaction for many knowledge-work tasks (a live voice+screen assistant), but reliability and cost at scale are still unproven outside demos.
Signal to track: Adoption of these APIs inside real enterprise customer-service or field-service workflows (not consumer demos), with disclosed latency and error rates.

21.26 Autonomous Cybersecurity (AI-driven defense/response)

Classification: Developing **What it is:** AI agents that detect, triage, and in some cases autonomously respond to security threats: Google's "Big Sleep" agent found a real-world exploited vulnerability in 2024, and Microsoft Security Copilot and CrowdStrike Charlotte AI now offer agentic response capabilities.
Why watch it: Security is one of the few domains where both attacker and defender are racing to adopt agentic AI simultaneously: falling behind here has asymmetric downside compared to most other items on this list. **Signal to track:** Whether your Security Operations Center (SOC)/EDR vendor's "autonomous response" features are enabled with real remediation authority in your environment, versus running in detect-and-recommend mode only.

21.27 Machine Customers (Gartner's term)

Classification: Developing **What it is:** AI agents that shop, negotiate, and buy on behalf of a human or organization: Gartner projected in 2023 that 20% of inbound customer-service contact volume would come from machine customers by 2026. **Why watch it:** If even partially realized, this changes customer-service design (agents don't need the same UX as humans) and demand-forecasting assumptions: worth checking against Gartner's own actual-vs-predicted reporting now that 2026 has arrived. **Signal to track:** Gartner's updated (post-hoc) assessment of whether the 20% figure was reached, and whether your own contact-center logs show a rising share of API/agent-originated (vs. human) interactions. (gartner.com)

21.28 Agent-to-Agent Commerce

Classification: Highly Uncertain/Speculative **What it is:** Transactions initiated and completed between two AI agents (a buying agent and a selling agent) with no human approving the specific transaction: the combination of agent discovery (A2A/MCP), payments (AP2/ACP), and reputation systems above. **Why watch it:** This is the composite, furthest-out scenario on this list: each prerequisite protocol exists individually in 2026, but no credible reporting yet shows agent-initiated B2B or B2C transactions happening at meaningful volume without a human confirming the specific purchase. **Signal to track:** The first disclosed enterprise procurement transaction completed end-to-end by autonomous agents on both sides, verified by something more than a vendor press release.

PUTTING IT INTO PRACTICE

The 90-Day Executive AI Learning Plan

Last updated September 11, 2026 · 10 min read · https://erikcaldwell.com/field-guide/ninety-day-executive-ai-learning-plan/

This plan sequences the topic catalog into a working study schedule. It assumes 3-5 hours per week: enough to read each week's topic entries closely (they are already written at executive length, roughly 10-15 minutes each), go deeper on one or two primary sources, run a small experiment, and sit with one hard question. Topics are referenced by their catalog number (Domain.Topic); go back to the full entry in the catalog for definitions, sources, and the complete list of questions to ask your team.

Treat the week numbers as a sequence, not a calendar. If Week 9 (Security) needs eight days instead of seven, take them: the plan is a priority order, not a countdown.

PHASE 1: FOUNDATIONS (Days 1-30 / Weeks 1-4)

Goal: build the shared vocabulary and mental models everything else depends on. By the end of this phase you should be able to sit in a vendor pitch or an architecture review and know which questions are the load-bearing ones.

Week 1: Strategy and the Operating Model

Topics: 1.1 AI Operating Models (Centralized/Federated/Hub-and-Spoke) · 1.2 AI Portfolio Management & Build vs Buy vs Configure · 1.3 Multi-Model Strategy & Vendor Dependency **Recommended Reading:** DORA AI Capabilities Model report (cited under 1.1): the single best evidence-based framework for what actually makes AI investment pay off. **Experiment:** List your organization's current AI initiatives on one page. For each, mark who owns the outcome and whether it has a kill date. If you can't fill in both columns for most rows, you've found your first governance gap. **Executive Question:** If I shut down every current AI pilot tomorrow, which ones would a business unit leader personally fight to keep, and why don't I already know the answer to that?

Week 2: What an Agent Actually Is

Topics: 2.1 What Is an AI Agent · 2.2 Agent Architectures · 2.6 Human Approval & Autonomy Levels · 2.7 Where Agents Beat vs Lose to Deterministic Software **Recommended Reading:** Anthropic's "Building Effective Agents" engineering post (cited under 2.2): the clearest primary-source explanation of when orchestration adds value versus overhead. **Experiment:** Pick one workflow your team automated with an agent (or wants to). Walk through it step by step and ask at each step: could this have been deterministic code instead? If the honest answer is yes for most steps, you have an agent-washing problem, not an AI strategy. **Executive Question:** For every agent we're running or planning, who has explicitly signed off on what level of autonomy it has, and can they show me that decision in writing?

Week 3: Interoperability and the Model Layer

Topics: 3.1 Model Context Protocol (MCP): Architecture & Enterprise Adoption · 3.2 MCP Security & Governance · 9.1 Frontier, Reasoning & Multimodal Models · 9.2 Open-Weight vs Proprietary Models **Recommended Reading:** The MCP first-anniversary and 2026 roadmap posts (cited under 3.1) for where the protocol is headed under Linux Foundation governance; skim one frontier-model announcement (cited under 9.1) to calibrate current capability. **Experiment:** Ask your platform team for a list of every MCP server currently connected to any internal system: internal or third-party. If nobody can produce that list in a day, you have a shadow-integration problem before you have an AI problem. **Executive Question:** Are we building a dependency on one model vendor's ecosystem, and if that vendor's pricing or terms changed tomorrow, what would it cost us to move?

Week 4: How Work Changes: Software and Knowledge Work

Topics: 4.1 The Software Engineering Continuum · 6.1 The Maturity Ladder (knowledge work) · 10.1 Prompt Engineering Fundamentals · 10.2 Context Engineering & System Architecture Around the Model **Recommended Reading:** DORA's 2025 State of AI-assisted Software Development report (cited under 4.1): read the velocity/stability tension section closely; it's the most important single finding in this document for a large engineering organization. **Experiment:** Ask five engineers and five knowledge workers, independently, where they'd place their own daily AI use on the two continuums in Topics 4.1 and 6.1. The spread in answers tells you more about your real adoption maturity than any dashboard. **Executive Question:** Is our organization optimizing for developers who feel faster, or for measurable delivery stability and business outcomes, and do we actually know if those are currently the same thing?

PHASE 2: ARCHITECTURE & TRANSFORMATION (Days 31-60 / Weeks 5-9)

Goal: go deep enough on the technical architecture, security posture, and workflow redesign questions that you can evaluate your own team's and vendors' proposals rather than rubber-stamping them.

Week 5: Agent Architecture, Deep Dive

Topics: 2.3 Long-Running, Computer-Use & Browser Agents · 2.4 Agent Memory & Persistent State · 2.5 Agent Sandboxes, Observability, Rollback & Idempotency · 4.2 Coding Agents & Autonomous Coding Agents · 4.3 Specification-Driven Development & Context Engineering for Code **Recommended Reading:** The OSWorld 2.0 benchmark paper (cited under 2.3) for a realistic read on current computer-use agent reliability: useful ammunition against vendor demos that only show the happy path. **Experiment:** Ask your engineering leadership what percentage of coding-agent output currently merges without human review versus with review. If nobody has that number, that's the finding. **Executive Question:** If one of our production agents took an irreversible action based on a bad decision, could we reconstruct exactly what it did and why, today, in under an hour?

Week 6: Software Engineering Operations and Workflow Redesign

Topics: 4.4 AI Code Review, Testing & Debugging · 4.6 Parallel Coding Agents, Supervision Ratios & Software Factories · 4.7 CI/CD & Quality Gates in an Agentic Development Model · 4.8 AI-Generated Code Risk & Secure Development Practices · 7.1 RPA vs BPM vs API Automation vs Agentic Workflows **Recommended Reading:** GitClear's code-quality research and the arXiv paper on security degradation in iterative AI code generation (both cited under 4.8): read these back to back; they're the sobering counterweight to velocity claims. **Experiment:** Pick one business process currently run as RPA or manual work. Have your team sketch what an agentic redesign would look like and where a deterministic step should stay deterministic on purpose. **Executive Question:** Are our quality gates (code review depth, test coverage requirements, architecture review) the same as they were two years ago, even though the volume and origin of code changed completely?

Week 7: Enterprise Knowledge and Data Architecture

Topics: 8.1 RAG, Embeddings & Vector Databases · 8.2 Hybrid Retrieval, Knowledge Graphs & GraphRAG (Monitor) · 8.3 Permissions-Aware Retrieval & Access Control Propagation · 8.4 Data Classification, Residency, Freshness & Provenance · 8.5 Enterprise, Personal & Agent Memory Architecture **Recommended Reading:** Anthropic's Contextual Retrieval post and the OWASP RAG Security Cheat Sheet (both cited under 8.1/8.3): the second one is the one to hand your CISO. **Experiment:** Ask your data team to demonstrate, live, that a search assistant correctly refuses to surface a document a test user isn't permissioned to see. If they can't demo it on request, don't trust that it works in production. **Executive Question:** When our RAG system is wrong, can we trace the answer back to the specific document it came from, and would that document have been visible to the user through normal permissions anyway?

Week 8: AI Platform Architecture and Evaluation

Topics: 11.1 AI Gateways & Model Gateways · 11.2 Centralized Policy Enforcement · 11.3 Gateway Architecture Choices (Monitor) · 12.1 AI Evals & Golden Datasets · 12.3 Hallucination, Groundedness & LLM-as-Judge · 12.4 Red Teaming & Continuous Production Evaluation **Recommended Reading:** OWASP Top 10 for LLM Applications, 2025 edition (cited under 12.4 and 13.1: read the evaluation-relevant items now, the security ones in Week 9). **Experiment:** Ask what percentage of your production AI features have a golden dataset and an automated eval that runs before every deploy. Most organizations at your scale will answer close to zero: that's the baseline you're trying to move. **Executive Question:** If a model provider silently updated the model behind one of our production features tomorrow, would we detect a quality regression before a customer or employee did?

Week 9: Security and Identity (this week runs long, budget extra time)

Topics: 13.1 OWASP Top 10 for LLM Applications · 13.2 Prompt Injection · 13.3 Excessive Agency & Goal Hijacking · 13.4 Memory/RAG Poisoning · 13.5 Insecure Output Handling · 13.6 MCP & Tool/Plugin Supply-Chain Risk · 13.7 Zero-Trust Architecture for Agents · 14.1 Non-Human Identity & Workload Identity · 14.2 Delegated Authorization & JIT Access · 14.3 Least Privilege, Transaction Limits & Audit Attribution **Recommended Reading:** Simon Willison's "lethal trifecta" post (cited under 13.2) and the

Invariant Labs GitHub MCP exploit writeup (cited under 13.6): both are short, concrete, and worth reading in full rather than summarized. **Experiment:** Ask your security team the single framing question that ties this whole domain together: "who authorized this AI to take this action, with this data, using these systems, at this time?": for your three highest-privilege agents. Silence or hand-waving is the finding. **Executive Question:** Do we have a non-human identity for every agent with write access to a production system, or are agents currently sharing service-account credentials with humans?

PHASE 3: SCALING THE ENTERPRISE (Days 61-90 / Weeks 10-13)

Goal: shift from "can we build this safely" to "are we running this as a managed capability": governance, workforce, measurement, and where to place your remaining strategic bets.

Week 10: Workforce Transformation and Knowledge Work at Scale

Topics: 5.1 Which Engineering Skills Appreciate vs Commoditize · 5.2 The Junior Engineer Pipeline Problem · 5.3 Changing Roles, Team Topology & Hiring Frameworks · 6.2 Enterprise AI Assistants & Enterprise Search · 6.3 Deep Research Agents & Document Generation (Monitor) · 6.4 Meeting Intelligence & Departmental Digital Workers **Recommended Reading:** Whatever your own engineering leadership can show you on junior-hire acceptance rates and time-to-productivity over the last two years (cited under 5.2): this is the topic where your own data matters more than any external source. **Experiment:** Ask how many entry-level engineering or analyst roles you've opened in the last 12 months versus three years ago. Then ask who decided the trend line, and whether anyone modeled what it does to your leadership pipeline in five years. **Executive Question:** If AI tools make experienced people more productive but reduce the number of junior roles that used to train the next generation of experienced people, who is accountable for closing that gap ten years out?

Week 11: Governance, Legal, and Compliance

Topics: 15.1 NIST AI Risk Management Framework & Generative AI Profile · 15.2 ISO/IEC 42001 · 15.3 EU AI Act · 15.4 AI, Intellectual Property & Copyright Risk · 15.5 AI System Inventory & Risk Classification **Recommended Reading:** NIST's AI RMF Playbook and the EU AI Act Service Desk implementation timeline (both cited under 15.1/15.3): the second one matters even if you have no EU operations today, because vendor compliance postures are converging on it. **Executive Question:** Do we have a single inventory of every AI system in production, who owns it, and what risk tier it's classified at, or would building that list today take weeks because it doesn't exist?

Week 12: Economics, Observability, and AI-Native Product Design

Topics: 16.1 Token Economics & Cost-Per-Outcome · 16.2 Cost Controls (routing, caching, substitution) · 16.3 Chargeback, Showback & Runaway Cost Risk (Monitor) · 17.1 LLMOps/AgentOps & Tracing Agent Trajectories · 17.2 Incident Response & Audit Replay · 18.1 Beyond the Chatbot · 18.2 Progressive Autonomy & Approval UX **Recommended Reading:** The FinOps Foundation's "Token Economics" material (cited under 16.1) and Nielsen Norman Group's "AI Agents as Users" research (cited under 18.1): the second one will change how you brief your product teams. **Experiment:** Pull last month's aggregate

model spend and ask your finance and platform teams to reconcile it against actual business outcomes, not usage volume. Expect this reconciliation to be harder than it should be: that gap is the finding. **Executive Question:** If one team's agent started running away (retrying, looping, or fanning out unexpectedly) would our cost controls catch it before finance did, three weeks later, in a monthly bill?

Week 13: Measurement, Adoption, and the Technology Radar

Topics: 19.1 Executive & Role-Based AI Literacy · 19.2 AI Champions & Communities of Practice · 19.3 Employee Displacement Fear & Change Management · 20.1 Engineering Metrics in the AI Era · 20.2 Knowledge-Worker Productivity Metrics · 20.3 Enterprise Value Metrics · 20.4 Why Prompt/Seat Counts Are Vanity Metrics · Domain 21, full read-through (Emerging Technology Radar) **Recommended Reading:** The NBER working paper on generative AI productivity gains by skill segment (cited under 20.2) (the single best corrective to flat "AI made us X% more productive" claims) and a full pass through the Executive Technology Radar table in this document. **Experiment:** Take your current AI dashboard, whatever it is, and for every metric on it ask: is this a proxy for an outcome, or is it a vanity metric dressed as one (seats provisioned, prompts sent, "AI-assisted" tickets closed)? Cut anything that fails that test. **Executive Question:** If I had to justify our entire AI investment to the board using only outcome metrics (revenue enabled, cost reduced, capacity released) and none of the adoption or activity metrics, could I do it convincingly today?

After Day 90

This plan gets you to fluency, not to done. Three habits carry it forward: revisit the Executive Technology Radar table quarterly and move items between Monitor, Experiment, and Adopt as evidence changes; re-score your organization against the Enterprise AI Capability Model every two quarters: maturity here moves in months, not years; and treat any Monitor-tier topic in the catalog (3.3, 4.5, 6.3, 9.5, 11.3, 12.2, 13.4, 13.5, 14.2, 15.2, 16.3, 18.3) as a standing item to re-read once it starts showing up in vendor pitches or team escalations, rather than something to schedule proactively.

PUTTING IT INTO PRACTICE

The 25 Concepts I Would Learn First

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/twenty-five-concepts-i-would-learn-first/

If time only permits 25 items from this entire document, this is the ranked list. It cuts across domains deliberately: the highest-leverage concepts for an executive are not evenly distributed by department, and security, architecture, and measurement concepts outrank several strategy concepts because getting them wrong is harder to reverse.

1. The agent loop. An AI agent perceives its environment, reasons about what to do, takes an action through a tool, and observes the result: repeating until the task is done or it stops itself. *Ranks #1 because every other agent concept in this document (memory, autonomy levels, sandboxing, cost) is a modification of this one loop, and most confusion about "what agents can and can't do" comes from not having this model in your head.*

2. Context engineering, not prompt engineering. The discipline of designing the whole system around a model (what data it retrieves, what tools it can call, what history it carries) because a model's output quality is dominated by what surrounds it, not by prompt wording. *Ranks #2 because it reframes nearly every AI initiative from "which prompt is best" to "what system did we build," which is the correct frame for architecture and investment decisions.*

3. Model Context Protocol (MCP). An open standard, now governed by the Linux Foundation's Agentic AI Foundation, for connecting AI models to external tools and data sources in a consistent way. *Ranks #3 because it has become the default integration layer across the industry in under two years: you will encounter it in nearly every vendor conversation, and it carries its own security surface you need to know exists.*

4. The software engineering continuum. Autocomplete → coding assistant → coding agent → agent-directed engineering → autonomous software factory: a progression of how much of the engineering task the AI owns versus the human. *Ranks #4 because it's the single frame that tells you where your organization actually sits today versus where a vendor pitch is implying you should be.*

5. Human approval and autonomy levels. The explicit, documented decision of how much an agent is allowed to do without a human checking first: from suggest-only to fully autonomous. *Ranks #5 because it is the primary governance lever you actually control, and most AI incidents trace back to an autonomy level nobody deliberately chose.*

6. Retrieval-Augmented Generation (RAG). Giving a model access to your organization's own documents and data at query time, rather than relying only on what it learned in training. *Ranks #6 because it's the mechanism behind almost every "AI that knows our business" product you'll evaluate, and its failure modes (stale data, wrong permissions, bad retrieval) are where most enterprise AI complaints originate.*

7. Permissions-aware retrieval. Ensuring an AI system only surfaces information the requesting user was already allowed to see: access control has to be enforced at retrieval time, not assumed. *Ranks #7 because getting this wrong is a data breach with an AI-shaped excuse, and it's the security gap least visible in a*

vendor demo.

8. Non-human / workload identity. Giving each AI agent its own verifiable identity (distinct from a shared service account or a human's credentials) so its actions can be authorized and audited individually. *Ranks #8 because "which agent did this" is unanswerable without it, and it's the prerequisite for nearly every other security and governance control in this document.*

9. Least privilege and audit attribution for agents. Limiting what an agent can do to the minimum required, with hard transaction limits, and keeping a record of what it did and why. *Ranks #9 because this is where "who authorized this AI to take this action, with this data, at this time" gets answered, or doesn't, when an incident happens.*

10. Prompt injection. An attack where instructions hidden in content an AI processes (a document, an email, a webpage) hijack its behavior: direct if the user does it, indirect if a third party embeds it in data the AI later reads. *Ranks #10 because it is the most common and least solved security problem in production LLM systems today, and no vendor has "fixed" it: only mitigated it.*

11. Excessive agency / goal hijacking. An agent given too much autonomy or too many tool permissions pursues its interpretation of a goal in ways nobody intended or authorized. *Ranks #11 because it's the agent-specific failure mode that traditional application security doesn't have a playbook for.*

12. AI evals and golden datasets. A curated, versioned set of test cases with known-correct answers, run automatically to check whether an AI system's outputs are still good before and after every change. *Ranks #12 because without this, you have no way to know if a model update, a prompt change, or a new feature made things better or worse: you're flying on vibes.*

13. Hallucination and groundedness. A model can produce fluent, confident, and completely wrong output; groundedness measures whether an answer is actually supported by the retrieved source material. *Ranks #13 because "it sounded right" is not a quality bar, and every AI-native product decision about citations, confidence, and human review traces back to this problem.*

14. Token economics and cost-per-outcome. Thinking about AI cost in terms of tokens processed is the wrong unit; the right unit is cost per successfully completed business outcome, which can vary by orders of magnitude across implementations of the "same" feature. *Ranks #14 because it's the difference between a FinOps conversation that controls spend and one that just watches a number go up.*

15. AI gateway / model gateway. A centralized layer that all AI traffic passes through, enforcing authentication, rate limits, data-loss prevention, content filtering, and cost controls consistently, regardless of which team or model is behind it. *Ranks #15 because it's the architectural choice that determines whether your governance policies are enforced everywhere or only in the systems someone remembered to configure.*

16. Multi-model strategy and vendor dependency. Deliberately architecting so you can swap or run multiple model providers, rather than hard-coding a dependency on one vendor's API, pricing, and roadmap. *Ranks #16 because model capability and pricing are both moving fast, and single-vendor lock-in compounds risk you can avoid with modest up-front design discipline.*

17. Build vs. buy vs. configure. The recurring decision, per use case, between building custom, buying a vendor product, or configuring a platform around your own data, and knowing that coding agents are shifting this calculus in ways that need scrutiny, not just enthusiasm. *Ranks #17 because it's the decision you'll be asked to make dozens of times, and getting the framework right once saves re-litigating it every time.*

18. DORA's AI-era engineering metrics and the velocity/stability tension. Research showing AI coding assistance can increase throughput while simultaneously straining delivery stability: the two don't automatically move together. *Ranks #18 because "engineers say they're faster" is not the same claim as "we ship more reliably," and conflating them is the most common measurement mistake in AI-assisted engineering.*

19. Skill-segmented productivity gains. Field research shows AI tools help less-experienced workers close skill gaps significantly more than it helps already-expert workers: productivity gains are not a flat percentage applied evenly across your workforce. *Ranks #19 because it changes how you should think about training investment, tool rollout sequencing, and what "average productivity lift" claims are actually hiding.*

20. Vanity metrics vs. outcome metrics. Seats provisioned, prompts sent, and "AI-assisted" tickets closed measure activity, not value; revenue enabled, cost reduced, and capacity released measure value. *Ranks #20 because nearly every AI dashboard you'll be shown defaults to the easy-to-measure activity metrics, and it's your job to insist on the harder outcome ones.*

21. Agent memory and persistent state. Unlike a single chatbot exchange, an agent that operates over hours or days needs a designed way to remember what it already tried, decided, or learned, and that memory can be poisoned or corrupted just like any other data store. *Ranks #21 because "it's just a chatbot" thinking badly underestimates the data-governance surface a persistent agent actually has.*

22. The knowledge-worker maturity ladder. Chat assistants → connected assistants → workflow automation → autonomous agents → autonomous business processes: a progression parallel to the software engineering continuum, for non-technical work. *Ranks #22 because it gives you the same "where do we actually sit" clarity for your knowledge-worker population that the SWE continuum gives you for engineering.*

23. NIST AI Risk Management Framework (Govern-Map-Measure-Manage). A voluntary, non-regulatory structure for identifying and managing AI risk across a system's lifecycle, increasingly treated as reference vocabulary by regulators and auditors even where it isn't mandatory. *Ranks #23 because it gives you a shared language with your legal, security, and audit functions that doesn't require waiting for binding regulation to be useful.*

24. The EU AI Act. Risk-tiered regulation (unacceptable, high-risk, limited, minimal) with a phased implementation timeline now extending into 2027-2028 for high-risk obligations: relevant even without EU operations because vendor compliance postures are converging toward it globally. *Ranks #24 because "we have no EU presence" is not the same as "this doesn't affect our vendor contracts and product roadmap."*

25. Zero-trust for agents and continuous red teaming. Treat every agent as potentially compromised and verify its actions continuously, rather than trusting it once it's inside your perimeter; test this assumption adversarially and on an ongoing basis, not just before launch. *Ranks #25 because agentic systems change*

behavior over time in ways a one-time security review can't catch, and this is the mindset shift that keeps governance from going stale.

PUTTING IT INTO PRACTICE

Executive Technology Radar

Last updated September 11, 2026 · 4 min read · https://erikcaldwell.com/field-guide/executive-technology-radar/

This table converts the topic catalog and the emerging-technology radar into a single decision instrument. **Current Importance** and **2-3 Year Potential** are independent judgments: some items are already important but plateauing, others are unimportant today but worth positioning for. **Recommended Action** is the operative column, and it is deliberately not "Adopt" by default:

- **Adopt**: fund and deploy now; the evidence and the risk profile both support it.
- **Build Capability**: not a single deployment decision, but an organizational muscle to build deliberately over the next 1-2 years (platform, governance, or skills).
- **Experiment**: worth a bounded pilot with a defined kill date, not yet worth enterprise-wide commitment.
- **Understand**: you need working knowledge and a point of view, but no near-term investment decision is required.
- **Monitor**: track quarterly; re-evaluate if the evidence changes, but do not staff or fund against it yet.
- **Ignore for Now**: genuinely not relevant to this organization's current business at this time; revisit only if your business model changes.

Topic	Current Importance	2-3 Year Potential	Recommended Action
Enterprise AI operating model (hub-and-spoke design)	High	High	Build Capability
AI portfolio management & build/buy/configure discipline	High	High	Build Capability
Multi-model strategy & vendor lock-in mitigation	Medium	High	Build Capability
Core agentic AI architecture (loops, orchestration, tool use)	High	High	Adopt
Agent autonomy levels & human approval gates	High	High	Build Capability
Model Context Protocol (MCP)	High	High	Adopt
Agent2Agent (A2A) protocol	Low	Medium	Monitor
Coding agents / agent-directed engineering	High	High	Adopt
Autonomous software factories (fleets, minimal human review)	Low	Medium	Experiment

Topic	Current Importance	2-3 Year Potential	Recommended Action
AI-generated code security risk & secure development controls	High	High	Build Capability
Engineering skill-mix shift & junior pipeline redesign	Medium	High	Understand
Enterprise AI assistants & enterprise search	High	High	Adopt
Agentic workflow automation vs. RPA/BPM	Medium	High	Experiment
RAG & enterprise knowledge retrieval	High	High	Adopt
Permissions-aware retrieval & data governance	High	High	Build Capability
Frontier & reasoning models (as a consumer, not a builder)	High	High	Understand
Open-weight models	Medium	Medium	Experiment
Context engineering as a system-design discipline	High	High	Build Capability
AI gateway / model gateway architecture	Medium	High	Build Capability
AI evaluation & golden datasets	Medium	High	Build Capability
Hallucination & groundedness controls	High	High	Build Capability
OWASP LLM Top 10 & prompt-injection defense	High	High	Adopt
Agent non-human identity & least privilege	Medium	High	Build Capability
NIST AI RMF / ISO 42001 alignment	Medium	High	Build Capability
EU AI Act compliance readiness	Low-Medium	High	Understand
AI IP & copyright risk management	Medium	Medium	Understand
Token economics & cost governance	Medium	High	Build Capability
AgentOps / observability & incident response	Medium	High	Build Capability
AI-native product UX (progressive autonomy, citations)	Medium	High	Experiment
AI champions & adoption change management	High	High	Adopt
Outcome-based AI value measurement	High	High	Build Capability
Autonomous software development (issue-to-PR-to-deploy, no human in loop)	Low	Medium	Experiment
Self-healing software	Low	Medium	Monitor

Topic	Current Importance	2-3 Year Potential	Recommended Action
Agent marketplaces, discovery, reputation & identity standards	Low	Medium	Monitor
Agent payments / agentic commerce (incl. agent-to-agent)	Low	Medium	Monitor
Persistent digital workers & autonomous departments	Low	Medium	Understand
Synthetic organizations	Low	Low/Uncertain	Ignore for Now
AI-native ERP & CRM	Low-Medium	Medium	Monitor
Generative UI & software generated on demand	Low	Medium	Monitor
Natural-language programming ("vibe coding" for non-developers)	Low-Medium	Medium	Experiment
Edge AI & local/on-device models	Low	Medium	Monitor
Embodied AI & robotics	Low	Low (for this business)	Ignore for Now
World models & AI-enhanced digital twins	Low	Uncertain	Ignore for Now
Synthetic data for training/eval	Low	Medium	Monitor
AI for scientific discovery	Low	Low (for this business)	Ignore for Now
Real-time multimodal agents (voice + vision + action)	Low	Medium	Monitor
Autonomous cybersecurity (AI-driven defense/response)	Medium	High	Experiment
Machine customers (Gartner's term)	Low	Medium/Uncertain	Monitor

A pattern worth naming explicitly: nothing on this list scores "Adopt" purely because it's new. Every Adopt row already has production-grade tooling, a body of evidence, and organizations your size running it successfully. Several genuinely important developments (autonomous departments, agent-to-agent commerce, embodied AI) sit at Monitor or Ignore for Now not because they're unimportant in the abstract, but because the evidence for enterprise deployment at your scale doesn't exist yet. Funding them now would be funding a bet, not a capability.

PUTTING IT INTO PRACTICE

The Enterprise AI Capability Model

Last updated September 11, 2026 · 7 min read · https://erikcaldwell.com/field-guide/enterprise-ai-capability-model/

Use this as a scoring instrument, not a reading list. Score your organization honestly against each of the 15 capability areas below. Most organizations at this scale sit at Level 1 or Level 2 in most areas in September 2026, and that is a normal starting point, not a failing grade. The point of the model is to see which areas are furthest behind relative to where your strategy actually needs them, so investment goes where it matters rather than where it's easiest to show progress.

Level 1: Experimental: Ad hoc, individual-initiative, no shared standard. **Level 2: Managed:** Deliberate for the highest-visibility cases, inconsistent elsewhere. **Level 3: Scaled:** Consistent standard practice across the organization. **Level 4: AI-Native:** The capability is fully embedded and largely automated; it is simply how the organization operates.

---

1. Strategy & Operating Model

- **L1 Experimental:** AI initiatives are ad hoc pilots run by individual teams with no central visibility or prioritization; there is no owned portfolio.
- **L2 Managed:** A central function tracks all AI initiatives, applies a build/buy/configure framework, and pilots have defined owners and kill criteria.
- **L3 Scaled:** AI investment is planned as part of the regular budget cycle under a deliberate hub-and-spoke operating model; multi-model vendor strategy is intentional, not accidental.
- **L4 AI-Native:** AI capability decisions are indistinguishable from core technology strategy; build/buy/configure calls are made against live cost and capability data, not annual planning cycles.

2. AI Platform & Gateway Architecture

- **L1:** Teams call model APIs directly and independently; no shared gateway, no consistent access control.
- **L2:** A gateway exists for at least the highest-risk use cases, enforcing basic authentication and rate limits.
- **L3:** All production AI traffic flows through a common gateway enforcing policy, cost limits, and data-loss prevention consistently.
- **L4:** The platform performs dynamic model routing, fallback, and cost-aware model selection automatically, with governed self-service onboarding for new use cases.

3. Data & Knowledge Architecture

- **L1:** AI features query raw data sources directly with no retrieval architecture or designed-in permissions enforcement.
- **L2:** RAG is deployed for flagship use cases; permissions are enforced but inconsistently across systems.
- **L3:** Permissions-aware retrieval, data classification, and source provenance are standard requirements for any new AI feature touching enterprise data.
- **L4:** A unified, governed knowledge layer serves every AI system consistently, with automatic staleness checks and access-control validation built in.

4. Software Engineering

- **L1:** Developers use AI autocomplete or chat individually; no organizational measurement of impact.
- **L2:** Coding assistants are standard-issue; some teams pilot coding agents for defined tasks under full human review.
- **L3:** Coding agents operate under defined supervision ratios; CI/CD quality gates and architecture validation have been rebuilt for AI-generated code volume.
- **L4:** Agent-directed engineering is standard for well-specified work; supervised software factories run fleets of agents against automated quality gates, and engineers have re-skilled toward specification, review, and system design.

5. Workforce Productivity (Knowledge Workers)

- **L1:** Employees use general-purpose chat assistants informally; no enterprise assistant or search is deployed.
- **L2:** An enterprise AI assistant and enterprise search are deployed organization-wide with basic adoption tracking.
- **L3:** Function-specific digital workers (HR, finance, legal, sales, procurement) are deployed for defined workflows with measured task completion.
- **L4:** Multi-step business processes run with agentic assistance end-to-end; humans supervise outcomes rather than perform the underlying tasks.

6. Agents & Automation

- **L1:** "Agent" is used loosely for any AI feature; no shared architecture or autonomy-level framework exists.
- **L2:** Agents are deployed for narrow, well-bounded tasks, with human approval required for any consequential action.

- **L3:** An explicit autonomy-level framework governs every production agent; sandboxing, rollback, and idempotency are standard requirements, not afterthoughts.
- **L4:** Agents coordinate across multiple workflows with autonomy calibrated per task type; the organization can state with confidence where agents outperform deterministic software for its own use cases and where they don't.

7. Security

- **L1:** AI systems are covered only by general application security practice; no LLM-specific threat model exists.
- **L2:** The OWASP Top 10 for LLM Applications has been reviewed; prompt-injection defenses exist for the highest-exposure systems.
- **L3:** Zero-trust principles are applied specifically to agents; MCP and tool/plugin supply-chain risk is actively managed; AI red teaming happens on a regular cadence.
- **L4:** Continuous adversarial testing runs against production AI systems; the security posture assumes any agent may be compromised and is designed to contain the blast radius by default.

8. Identity (Agent & Non-Human Identity)

- **L1:** Agents share service-account credentials with humans or with each other; no distinct agent identity exists.
- **L2:** High-risk agents have distinct credentials, but delegation and scoping are managed manually, case by case.
- **L3:** A non-human identity framework issues every agent its own identity with least-privilege scoping and transaction limits, enforced through delegated, JIT authorization.
- **L4:** Agent identity, authorization, and audit trails are fully standardized platform-wide; any agent's authority can be traced, scoped, and revoked in real time.

9. Governance, Legal & Compliance

- **L1:** No formal AI risk framework or system inventory exists; compliance is handled reactively, case by case.
- **L2:** An AI system inventory exists for the highest-visibility systems, with informal risk classification.
- **L3:** NIST AI RMF or ISO/IEC 42001 principles are formally adopted; every production AI system is inventoried and risk-classified; EU AI Act exposure has been assessed regardless of EU footprint.
- **L4:** AI governance is embedded in standard technology governance processes, with automated system registration, risk classification, and compliance monitoring across the full portfolio.

10. Evaluation & Quality Engineering

- **L1:** AI feature quality is assessed informally, by demo or anecdote, with no systematic testing.
- **L2:** Golden datasets and basic evals exist for the highest-profile AI features only.
- **L3:** Evals run automatically before every production deployment; hallucination/groundedness checks and trajectory evaluation are standard for agentic systems; red teaming happens on a defined schedule.
- **L4:** Continuous production evaluation runs alongside live traffic, catching quality regressions (including silent model updates from vendors) before they reach end users at scale.

11. Observability & AgentOps

- **L1:** AI system behavior is opaque; failures are diagnosed manually and inconsistently.
- **L2:** Logging exists for the highest-risk agents, but there is no standardized tracing across systems.
- **L3:** LLMOps/AgentOps tooling traces agent trajectories consistently; incident-response playbooks exist for AI-specific failures, with audit replay capability.
- **L4:** Full-fidelity tracing, alerting, and audit replay are standard across every production agent, integrated with the organization's broader observability stack.

12. AI FinOps & Economics

- **L1:** AI spend is tracked only at the vendor-invoice level, with no per-team or per-use-case visibility.
- **L2:** Cost is tracked per major initiative, but chargeback or showback is informal or absent.
- **L3:** Token economics are understood in cost-per-outcome terms, not just cost-per-token; routing, caching, and model substitution are used deliberately; chargeback/showback is standard practice.
- **L4:** Real-time cost governance catches runaway agent spend automatically; cost-per-outcome is reviewed alongside every other business unit's financials as a matter of course.

13. Change Management & Adoption

- **L1:** AI tools are introduced with no formal change management; adoption is left to individual initiative.
- **L2:** Basic role-based AI literacy training exists; an acceptable use policy has been published.
- **L3:** AI champions and communities of practice operate across business units; a use-case library captures and spreads what's working; displacement fears are addressed directly rather than avoided.
- **L4:** AI fluency is a standard part of role competency expectations organization-wide; adoption is measured and managed like any other major change program, with continuous feedback loops.

14. AI-Native Product Development

- **L1:** AI features are bolted onto existing chatbot-style interfaces with no distinct design discipline.
- **L2:** Some products incorporate citations or confidence indicators for AI-generated content, inconsistently.
- **L3:** Progressive autonomy and approval UX are designed deliberately; failure states, provenance, and confidence indicators are first-class design considerations across products.
- **L4:** Products are designed AI-native from the outset (including, where relevant, treating AI agents as users of the product's own interfaces) with autonomy and trust calibrated by design rather than added afterward.

15. Measurement & Business Value

- **L1:** AI success is reported through activity metrics: seats provisioned, prompts sent, "AI-assisted" volume.
- **L2:** Some engineering metrics (DORA-style) are tracked for AI-assisted development, but knowledge-worker and enterprise value metrics remain activity-based.
- **L3:** Outcome metrics (revenue enabled, cost reduced, capacity released) are tracked for major initiatives, and skill-segmented productivity effects are understood rather than assumed uniform.
- **L4:** AI value measurement is integrated into standard business reporting; every material AI investment has an owned outcome metric the organization would defend to its board without relying on adoption or usage numbers.

Using this model: Score each area today, then again every two quarters: maturity here moves in months, not years, in either direction. Prioritize investment in whichever areas are most behind relative to your strategy's actual dependencies, not the areas easiest to show a Level 4 slide about. A common and costly failure pattern is racing to Level 3-4 in Workforce Productivity or AI-Native Product Development while Security, Identity, and Governance sit at Level 1: visible progress with an invisible, compounding liability underneath it.

REFERENCE

References

Last updated September 11, 2026 · 6 min read · https://erikcaldwell.com/field-guide/references/

Every source cited across the topic catalog, consolidated and alphabetized. Sources are primary documentation, standards bodies, peer-reviewed or preprint research, and named engineering organizations' own published data: not marketing content, SEO content, or generic consultant thought leadership. This list is a lookup appendix; the in-context citation for each source appears under the relevant topic's "Recommended Learning" entry in the catalog.

- A Timeline of Model Context Protocol (MCP) Security Breaches
- A2A and MCP: A2A Protocol
- A2A Protocol Surpasses 150 Organizations...: Linux Foundation
- Access Foundry Models and Other Language Models Through a Gateway: Azure Architecture Center
- Activating your internal AI champions (GitHub)
- Agent Authority Least Privilege Framework: FINOS
- Agentic AI: Threats and Mitigations
- AgentOps: Enabling Observability of LLM Agents (arXiv)
- AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases
- agents.md: the open standard
- AI Act: Shaping Europe's digital future (European Commission)
- AI Agents as Users: Nielsen Norman Group
- AI Agents Have an Authorization Problem, Not Just an Identity Problem
- AI Chatbots Discourage Error Checking: Nielsen Norman Group
- AI gateway capabilities in Azure API Management
- AI Red Teaming Initiative: OWASP Gen AI Security Project
- AI Risk Management Framework (NIST)
- AI RMF: Generative Artificial Intelligence Profile (NIST)
- Announcing Microsoft Entra Agent ID
- Anthropic: Claude Code best practices
- Anthropic: Effective harnesses for long-running agents
- Anthropic: How we built our multi-agent research system
- appleinsider.com
- Article 4: AI literacy: EU Artificial Intelligence Act
- Auditing and Logging AI Agent Activity: A Guide for Engineers: LoginRadius Engineering Blog
- Authorization and Governance for AI Agents: Runtime Authorization Beyond Identity at Scale
- Bersin

- BIRD-SQL benchmark
- Block denied topics to help remove harmful content: Amazon Bedrock
- Building Effective AI Agents: Anthropic
- Canaries Dashboard (Stanford Digital Economy Lab)
- Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of AI (Stanford Digital Economy Lab)
- Chargeback vs. Showback: Cloud Cost Allocation Models Explained: CloudZero
- Claude Enterprise
- Claude prompting best practices (Claude Platform Docs)
- Cloud Security Alliance: The Vibe Coding Governance Gap
- cloud.google.com
- Computer use tool: Claude Platform Docs
- Computer-Using Agent: OpenAI
- Contextual Retrieval (Anthropic)
- deloitte.com
- Deloitte: The State of AI in the Enterprise
- developers.googleblog.com
- Diving Into the MCP Authorization Specification: Descope
- Donating the Model Context Protocol and establishing the Agentic AI Foundation: Anthropic
- DORA 2025: State of AI-assisted Software Development
- DORA AI Capabilities Model report
- DORA Insights
- DORA research on AI
- DORA: Balancing AI tensions: moving from adoption to effective SDLC use
- Effective context engineering for AI agents: Anthropic
- en.wikipedia.org
- Enterprise AI Operating Model: Hub-and-Spoke, Federated, or Centralized?
- EU agrees Digital Omnibus deal to simplify AI rules: White & Case
- EU AI Act implementation timeline: AI Act Service Desk
- Explainable AI in Chat Interfaces: Nielsen Norman Group
- Factory.ai: From coding agents to software factories
- Fine-tuning (OpenAI Developers guide)
- FinOps for AI Overview: FinOps Foundation
- Forrester
- Gartner
- Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026
- gartner.com

- Gemini 3: Introducing the latest Gemini AI model (Google)
- Generative AI at Work: Brynjolfsson, Li, Raymond (NBER Working Paper 31161)
- Generative Artificial Intelligence and Copyright Law: Congressional Research Service
- GitClear: AI Copilot Code Quality 2025 research
- GitHub MCP Exploited: Accessing private repositories via MCP
- GitHub Octoverse 2025
- GitHub Octoverse 2025
- Glean
- Glean
- Google Cloud
- Google: NotebookLM Adds Deep Research
- GPT-5.1 System Card Addendum (OpenAI)
- GraphRAG: Unlocking LLM discovery on narrative private data (Microsoft Research)
- HHEM v2: A New and Improved Factual Consistency Scoring Model: Vectara
- How we built our multi-agent research system: Anthropic
- Hybrid search (Weaviate Documentation)
- Identifying Token Costs Hiding in Your Agentic Loop: MachineLearningMastery
- InfoQ: Agentic fitness functions: extending evolutionary architecture beyond deterministic rules
- InfoWorld: How Google is using LLMs for complex internal code migrations
- Inside the LLM Call: GenAI Observability with OpenTelemetry: OpenTelemetry Blog
- Introducing Claude Opus 4.5 (Anthropic)
- Introducing the Model Context Protocol (Anthropic)
- Introduction: Model Context Protocol
- ISO/IEC 42001 explained
- ISO/IEC 42001:2023: AI management systems
- ISO/IEC 42001:2023: Microsoft compliance overview
- Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena (arXiv)
- learn.microsoft.com
- Least privilege for AI agents with Microsoft Entra Agent ID
- Least privilege for AI agents: Identity, access, and tool binding
- Levels of Autonomy for AI Agents: Knight First Amendment Institute
- Leveraging model distillation to fine-tune a model (OpenAI Cookbook)
- Llama 4 Community License Agreement
- LLM Agent Evaluation Metrics: Tool Calling, Task Completion, Reasoning, and Trace-Based Evals: Confident AI
- LLM Evaluation: Best Practices and Methods: Databricks Engineering Blog
- LLM Routing and Model Cascades: How to Cut AI Costs Without Sacrificing Quality

- LLMOps Guide: Build Fast, Cost-Effective LLM Apps: Redis Engineering Blog
- MCP Authorization specification
- MCP Security Notification: Tool Poisoning Attacks: Invariant Labs
- MCP Tool Poisoning: OWASP Foundation
- Measuring AI agent autonomy in practice: Anthropic
- MemGPT: Towards LLMs as Operating Systems (arXiv)
- Memory for Claude Managed Agents (Anthropic)
- merics.org
- Meta's Llama license is still not Open Source (Open Source Initiative)
- METR: Measuring the impact of AI on experienced open-source developer productivity
- Microsoft Agent 365 Overview
- Microsoft GraphRAG project page
- Microsoft Learn: Overview of Process Mining in Power Automate
- Microsoft Support: Use Researcher in Copilot Notebooks
- Microsoft: Catch Up on Meetings with Copilot in Teams
- Migrating Code At Scale With LLMs At Google (arXiv 2504.09691, FSE 2025)
- MIT NANDA
- MIT report: 95% of generative AI pilots at companies are failing
- Model router for Microsoft Foundry: concepts
- n8n Docs
- NIST AI 600-1: Artificial Intelligence Risk Management Framework: Generative AI Profile
- NIST AI RMF Playbook
- NSA/CISA: Model Context Protocol (MCP) Security
- One Year of MCP (Model Context Protocol blog)
- OpenAI
- OpenTelemetry GenAI Semantic Conventions: MLflow documentation
- OSWorld 2.0 (arXiv 2606.29537)
- Overview of model routing: Google Cloud API Gateway
- OWASP Gen AI Security Project: Resources
- OWASP LLM01:2025 Prompt Injection
- OWASP LLM04:2025 Data and Model Poisoning
- OWASP LLM05:2025 Improper Output Handling
- OWASP LLM06:2025 Excessive Agency
- OWASP LLM08:2025 Vector and Embedding Weaknesses
- OWASP Top 10 for LLM Applications 2025
- OWASP Top 10 for LLM Applications 2025
- pgvector (GitHub)

- Prompt engineering best practices for 2026 (Anthropic)
- Prompt injection: Wikipedia
- RAG Security Cheat Sheet (OWASP)
- RAG with Permissions (Supabase Docs)
- RAGAS: Automated Evaluation of Retrieval Augmented Generation (arXiv)
- ReAct: Synergizing Reasoning and Acting in Language Models
- Regulation (EU) 2024/1689 (EUR-Lex, official text)
- Remove PII from conversations by using sensitive information filters
- Replay: reconstructing an agent action from the audit log: Deixic
- research.google
- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (Lewis et al., 2020)
- Rising AI Adoption Spurs Workforce Changes (Gallup)
- Safety and content filters: Vertex AI
- salesforce.com
- salesforce.com
- Salesforce: Agentforce Developer Guide: Get Started
- Securing Agentic Applications Guide 1.0
- Security Degradation in Iterative AI Code Generation (arXiv 2506.11022, IEEE ISTAS 2025)
- Security, Guardrails, and Observability in Amazon Bedrock
- Shadow deployment vs. canary release of machine learning models
- Simon Willison's prompt-injection archive
- SPIFFE: Securing the identity of agentic AI and non-human actors
- State of UX 2026: Nielsen Norman Group
- stripe.com
- Structured Outputs (OpenAI API)
- Team Topologies as the "infrastructure for agency" with AI
- Team Topologies: the book's site
- Techzine
- The 2026 MCP Roadmap: Model Context Protocol Blog
- The AI Champion role (OpenAI Academy)
- The lethal trifecta for AI agents
- The New York Times v. Microsoft and OpenAI: case background and status
- The Non-Human Identity Governance Vacuum
- The Register, "AI coding tools make developers slower, study finds"
- The State of AI: Global Survey 2025 (McKinsey)
- The Vulnerable MCP Project
- Token Economics: The Atomic Unit of AI Value: FinOps Foundation

- Tool use with Claude: Claude Platform Docs
- TRAJECT-Bench: A Trajectory-Aware Benchmark for Evaluating Agentic Tool Use (arXiv)
- UiPath
- UiPath: What is Agentic Orchestration?
- Understanding intelligent prompt routing in Amazon Bedrock
- Understanding the context window (Anthropic docs)
- US Supreme Court declines to consider whether AI alone can create copyrighted works: Morgan Lewis
- vectara/hallucination-leaderboard (GitHub)
- When Should We Trust AI? Magic-8-Ball Thinking and AI Hallucinations: Nielsen Norman Group
- Workato: Agentic (docs)

REFERENCE

Glossary of Terms and Acronyms

Last updated September 11, 2026 · 9 min read · https://erikcaldwell.com/field-guide/glossary-of-terms-and-acronyms/

Every acronym and specialized term used in this document, defined in one place for quick lookup. Terms are also defined inline at their first use in the running text; this glossary is the reference copy.

- **A2A (Agent2Agent):** An open protocol, originally developed by Google and now governed by the Linux Foundation, that lets independent AI agents discover each other's capabilities and communicate to delegate and coordinate tasks.
- **ABAC (Attribute-Based Access Control):** An access-control model that grants permissions based on attributes of the user, resource, and context, rather than fixed roles.
- **ACP (Agentic Commerce Protocol):** An emerging standard for agent-initiated commerce transactions, alongside AP2, still consolidating as of this writing.
- **AI (Artificial Intelligence):** Used throughout this document as an umbrella term; specific mechanisms (agents, LLMs, RAG, and so on) are defined individually where precision matters.
- **AP2 (Agent Payments Protocol):** A protocol for agent-initiated payment transactions.
- **API (Application Programming Interface):** A defined interface that lets one piece of software call another's functionality directly, without a user interface.
- **APIM (API Management):** Infrastructure for managing, securing, and monitoring API traffic, including AI gateway traffic.
- **APM (Application Performance Monitoring):** Tooling that tracks the health, latency, and errors of running software systems.
- **AUP (Acceptable Use Policy):** A published policy telling employees what AI tools they may use, on what data, and under what constraints.
- **AWS (Amazon Web Services):** Amazon's cloud computing platform.
- **B2B (Business-to-Business):** Commerce or transactions conducted between businesses rather than with individual consumers.
- **B2C (Business-to-Consumer):** Commerce or transactions conducted between a business and individual consumers.
- **BCG (Boston Consulting Group):** A management consulting firm cited for AI value-tracking research.
- **BIRD-SQL (Big Bench for Large-Scale Database Grounded Text-to-SQL Evaluation):** A benchmark for evaluating a model's ability to translate natural language into correct SQL queries.
- **BM25 (Best Matching 25):** A keyword-ranking algorithm used in search and hybrid retrieval systems.
- **BPM (Business Process Management):** A discipline and toolset for modeling, automating, and improving structured business processes.

- **CD (Continuous Delivery/Deployment):** The practice of automatically preparing or releasing every validated code change to production; paired with CI as "CI/CD."
- **CFO (Chief Financial Officer):** The executive responsible for an organization's financial management.
- **CI (Continuous Integration):** The practice of frequently merging and automatically testing code changes; paired with CD as "CI/CD."
- **CISA (Cybersecurity and Infrastructure Security Agency):** The U.S. federal agency responsible for cybersecurity and infrastructure protection guidance.
- **CISO (Chief Information Security Officer):** The executive responsible for an organization's information security program.
- **CRM (Customer Relationship Management):** Software systems that manage an organization's interactions with current and prospective customers.
- **CUA (Computer-Using Agent):** An AI agent that acts directly on a computer's graphical interface or browser, viewing screenshots and issuing mouse/keyboard actions, rather than calling structured APIs.
- **CVE (Common Vulnerabilities and Exposures):** A public catalog of disclosed software security vulnerabilities, each with a unique tracking identifier (e.g., CVE-2025-6514).
- **DLP (Data Loss Prevention):** Controls that detect and block sensitive data from leaving an organization's systems, including through AI prompts and outputs.
- **DORA (DevOps Research and Assessment):** A Google-affiliated research program that produces the industry's leading evidence-based research on software delivery performance and, since 2024-2025, AI-assisted engineering.
- **EDR (Endpoint Detection and Response):** Security tooling that monitors endpoint devices for threats and can trigger automated response actions.
- **ERP (Enterprise Resource Planning):** Integrated software systems (finance, HR, supply chain, and similar) that run an organization's core operational processes.
- **EU (European Union):** The political and economic union whose AI Act is a primary regulatory reference point in this document.
- **FINOS (Fintech Open Source Foundation):** A Linux Foundation project developing open-source standards and frameworks for financial services technology, including agent governance frameworks.
- **FSE (Foundations of Software Engineering):** A leading academic software engineering conference, cited here for peer-reviewed research on AI-assisted code migration.
- **GCP (Google Cloud Platform):** Google's cloud computing platform.
- **GenAI (Generative AI):** AI systems that generate new content (text, code, images, and so on) rather than only classifying or scoring existing content.
- **GPAI (General-Purpose AI):** A regulatory category under the EU AI Act covering foundation models not built for a single narrow purpose.
- **GPT (Generative Pre-trained Transformer):** OpenAI's family of large language models.

- **GUI (Graphical User Interface):** A visual, point-and-click interface, as distinct from a command-line or programmatic interface.
- **HHEM (Hughes Hallucination Evaluation Model):** Vectara's open model for scoring whether a generated answer is factually grounded in its source material.
- **HR (Human Resources):** The organizational function responsible for workforce management.
- **HTTP (Hypertext Transfer Protocol):** The foundational protocol of the web, referenced here as a comparison point for newer agent-interoperability protocols.
- **IAM (Identity and Access Management):** Systems and policies that govern who (or what) can access which resources.
- **ID (Identity/Identifier):** Used in this document both generically and in named products (e.g., Microsoft Entra Agent ID).
- **IEC (International Electrotechnical Commission):** The international standards body that co-publishes standards such as ISO/IEC 42001 and ISO/IEC 27001 with the International Organization for Standardization.
- **IEEE (Institute of Electrical and Electronics Engineers):** A professional association that publishes technical standards and research, including the ISTAS conference cited in this document.
- **IP (Intellectual Property):** Legal rights in creative and inventive work; a live risk area for AI-generated content and training data.
- **ISO (International Organization for Standardization):** The international standards body behind standards such as ISO/IEC 42001 (AI management systems) and ISO/IEC 27001 (information security).
- **ISTAS (International Symposium on Technology and Society):** An IEEE-affiliated conference, cited here for research on security degradation in iterative AI code generation.
- **IT (Information Technology):** The organizational function responsible for technology infrastructure and systems.
- **JIT (Just-in-Time):** In this document, just-in-time access: granting elevated permissions only for the duration of a specific approved action.
- **JSON (JavaScript Object Notation):** A lightweight, structured data format commonly used for API requests, responses, and model output schemas.
- **KPI (Key Performance Indicator):** A measurable value used to track progress toward a goal; referenced generically in discussions of measurement discipline.
- **L1-L4 (Maturity Levels):** The four-stage maturity scale (Experimental, Managed, Scaled, AI-Native) used throughout the Enterprise AI Capability Model to score organizational capability.
- **LLM (Large Language Model):** A machine learning model trained on large volumes of text to generate and reason over natural language; the technology underlying most modern AI agents and assistants.
- **MCP (Model Context Protocol):** An open standard, now governed by the Linux Foundation's Agentic AI Foundation, for connecting AI models to external tools and data sources in a consistent way.

- **METR (Model Evaluation & Threat Research):** An independent research organization known for a randomized controlled trial measuring real-world AI coding productivity effects.
- **MIT (Massachusetts Institute of Technology):** Cited here as the affiliation behind the NANDA research initiative on enterprise AI pilot outcomes.
- **M365 (Microsoft 365):** Microsoft's cloud productivity suite (Office apps, Copilot, and related services).
- **MTTR (Mean Time to Restore):** A DORA software delivery metric measuring how long it takes to restore service after a failure.
- **NANDA (Networked Agents and Decentralized AI):** An MIT research initiative studying enterprise AI agent adoption and outcomes.
- **NBER (National Bureau of Economic Research):** A U.S. economic research organization, cited here for field-experiment research on AI productivity effects segmented by worker skill level.
- **NHI (Non-Human Identity):** A distinct, verifiable identity issued to an AI agent or automated workload, separate from any human user's credentials.
- **NIST (National Institute of Standards and Technology):** The U.S. federal agency behind the AI Risk Management Framework (AI RMF) and its Generative AI Profile, widely used as reference vocabulary for AI governance.
- **NLP (Natural Language Processing):** The field of computing concerned with processing and generating human language.
- **NSA (National Security Agency):** The U.S. federal agency that co-published, with CISA, security guidance on Model Context Protocol risk.
- **OAuth:** An open standard for delegated authorization, used (in its 2.1 revision) as the authorization model underlying several agent-to-agent and agent-identity protocols discussed in this document.
- **OWASP (Open Worldwide Application Security Project):** A nonprofit foundation that publishes widely adopted, community-developed application security guidance, including the LLM/Generative AI security Top 10 referenced throughout this document.
- **PDF (Portable Document Format):** A fixed-layout document file format, referenced here for source materials such as the NIST AI RMF.
- **PII (Personally Identifiable Information):** Data that can identify a specific individual, a primary concern for data loss prevention and privacy controls.
- **PR (Pull Request):** A request to merge a proposed code change into a shared codebase, subject to review.
- **RAG (Retrieval-Augmented Generation):** Giving a model access to an organization's own documents and data at query time, rather than relying only on what it learned in training.
- **RAGAS (Retrieval Augmented Generation Assessment):** An open-source framework for automatically evaluating RAG system output quality.
- **RBAC (Role-Based Access Control):** An access-control model that grants permissions based on a user's assigned role.

- **RMF (Risk Management Framework):** In this document, most often NIST's AI Risk Management Framework, a voluntary structure for identifying and managing AI risk across a system's lifecycle.
- **ROI (Return on Investment):** A measure of the value gained from an investment relative to its cost.
- **RPA (Robotic Process Automation):** Software "bots" that automate repetitive, rule-based digital tasks by mimicking user actions on existing interfaces.
- **SaaS (Software as a Service):** Software delivered and licensed as a hosted, subscription-based cloud service rather than installed on-premises.
- **SAP:** A major enterprise software vendor, referenced here for its Joule AI agents embedded across ERP functions.
- **SAST (Static Application Security Testing):** Automated analysis of source code to find security vulnerabilities without executing the program.
- **SDK (Software Development Kit):** A packaged set of tools, libraries, and documentation for building on a given platform or protocol.
- **SDLC (Software Development Lifecycle):** The end-to-end process of planning, building, testing, deploying, and maintaining software.
- **SEO (Search Engine Optimization):** Techniques for improving a page's visibility in search results; referenced in this document only as an excluded source-quality category.
- **SOC (Security Operations Center):** The team and systems responsible for monitoring and responding to security incidents.
- **SPIFFE (Secure Production Identity Framework for Everyone):** An open standard for issuing verifiable workload identities, increasingly extended to cover AI agents.
- **SPIRE:** The reference runtime implementation of the SPIFFE standard.
- **SQL (Structured Query Language):** The standard language for querying and managing relational databases.
- **UI (User Interface):** The means by which a person interacts with a software system.
- **US (United States):** Referenced primarily in the context of U.S. federal agencies and government reference frameworks (NIST, NSA, CISA).
- **UX (User Experience):** The overall experience and usability of a product or system from the user's perspective.
- **VM (Virtual Machine):** An isolated, software-based emulation of a computer system, commonly used to sandbox agent actions.

Additional Terms

- **Agent (AI agent):** A system that perceives its environment, reasons about what to do, takes an action through a tool, and observes the result, repeating until a task is complete.
- **Agentic AI:** AI systems built around the agent loop, as distinct from single-turn chat or classification systems.

- **Context engineering:** The discipline of designing the whole system around a model (what data it retrieves, what tools it can call, what history it carries), rather than focusing narrowly on prompt wording.
- **Frontier model:** The current highest-capability general-purpose models from the major AI labs.
- **Groundedness:** Whether a model's answer is actually supported by the retrieved source material, as distinct from output that is merely fluent and confident.
- **Hallucination:** Fluent, confident, but factually incorrect model output.
- **Hub-and-spoke (operating model):** An organizational pattern in which a central team owns shared platform and governance, while business-unit "spoke" teams own delivery.
- **Prompt injection:** An attack in which instructions hidden in content an AI processes hijack its behavior; direct if the user does it, indirect if a third party embeds it in data the AI later reads.
- **Zero-trust (for agents):** Treating every agent as potentially compromised and verifying its actions continuously, rather than trusting it once it is inside the organization's perimeter.

Changelog

The most recent versions of this document. The full history is at https://erikcaldwell.com/field-guide/changelog/.

Version	Date	Summary
v2026.09.1	2026-09-11	0 section(s) revised, 28 added, 0 renamed, 0 removed.