Palmetto Interactive

WHITEPAPER · AGENTIC AI SYSTEMS

The second mile



The demo works. What kills most enterprise agent projects is the distance between that demo and something anyone will sign off on. And that distance is architecture rather than model quality.

Matt Lucas · Founder & Principal, Palmetto Interactive
Twenty years across enterprise risk, finance and compliance technology, and enterprise AI delivery.

AUGUST 2026

01 · THE GAP, RESTATED

The demo was never the hard part

The question has moved. It’s no longer whether an agent can do the work in a demo. It’s why a demo that plainly works still dies before production.

Two data points frame this, and both deserve care rather than being quoted as slogans. Gartner predicts that over 40% of agentic AI projects get canceled by the end of 2027. They put it down to escalating costs, unclear business value, and inadequate risk controls. They also note something worth sitting with. Of the thousands of vendors claiming agentic capability, only around 130 are building anything of the kind.[1]

The more widely circulated figure comes from MIT’s NANDA initiative. It reported that 95% of generative AI pilots showed no measurable P&L return. The blame went to gaps in learning, integration, and contextual adaptation rather than model quality or talent.[2] That figure has been publicly disputed on methodology. Read it as a directional signal from a limited sample rather than a measurement. We cite it because its *explanation* matches what we see in the field. The number itself isn’t settled.

Both point at the same thing. These aren’t model failures. They’re the absence of the engineering that has to exist around a model before an accountable organization lets it act.

02 · THE DIAGNOSIS

Eight ways agent projects die

Ranked by how often they actually end deployments, rather than how interesting they are.

FAILURE MODE	WHAT IT LOOKS LIKE
01 · Launched on vibes	No representative test set, no regression detection, no release gate. The system quietly degrades after a prompt, model, or tool change, and nobody can prove when it started.
02 · Grounding failure	The agent pulls incomplete, stale, or conflicting evidence and acts on it with full confidence. Architecturally there’s no difference between "no answer" and "wrong answer."
03 · Tools as afterthought	Tools built for demo convenience get carried into production. Broad scopes, no argument validation, no idempotency. Blast radius scales with what the agent is allowed to do rather than with attacker sophistication.
04 · No observability	No per-step trace of what the agent read, decided, and called. Something goes wrong in week six and nobody can reconstruct it, so the system gets switched off rather than fixed.
05 · Cost blowup	Three cents a run across ten test cases becomes unbudgeted at fifty thousand runs a month. Retries, tool loops, and inter-agent chatter multiply the tokens. Nobody set a budget or an alert.
06 · Premature topology	Multi-agent orchestration adopted because it’s fashionable rather than because the task breaks down that way. Circular delegation, context lost between agents, quiet partial failures nobody aggregates.
07 · The accountability gap	No named human owns the consequence of a given action. Risk asks "who approved this," nobody can answer, and the system stalls in review indefinitely.
08 · Access fiction	The architecture assumes API access that really needs a six-week entitlement request, or doesn’t exist at record-level granularity. Found in week eight of a twelve-week pilot.

> *None of these is a model problem. Every one is a decision somebody put off.*

03 · TOPOLOGY

More autonomy is rarely the fix

When an agent is unreliable, the instinct is to give it more room. More agents, more delegation, more autonomy. That usually makes things worse. You multiply the places context can get lost and leave the original grounding problem untouched.

Default to a single agent with a well-scoped tool set and deterministic pre- and post-processing, wherever the task is single-threaded reasoning over bounded context. Most enterprise workflows have that shape. Service resolution, analyst briefing, controlled case handling, revenue operations.

Reach for multi-agent decomposition only when the task really has independent sub-problems with clean interfaces. Or when a distinct specialist role materially improves quality, like a critique pass separated from a generation pass. Then justify it in writing before you build it. Regulated buyers respond well to "we considered multi-agent here and rejected it, and here’s why."

04 · THE CONTEXT LAYER

Grounding, provenance, and the honest "I don't know"

Every retrieved fact should carry its source, timestamp, and confidence, and the output should cite what it actually used. That much is widely accepted now.

The part people skip is harder. **"Insufficient evidence" and "conflicting evidence" have to be first-class outputs**, rather than conditions the agent papers over. Your workflow needs a defined next step for both. Escalate, ask, or decline. An agent with no way to say "I can’t answer this" will always answer, and it’ll answer confidently.

This is where a governed semantic layer earns its keep for agentic work specifically. Shared entity identity and explicit relationships cut the retrieve-and-guess failure mode directly, because the agent walks structure rather than inferring it. Every agentic engagement doesn’t need a knowledge graph. The ones whose evidence spans systems that disagree usually do.

05 · TOOLS AND PERMISSIONS

Least privilege has to be architecture

Prompt injection into an over-permissioned agent is the top risk category in OWASP's list for LLM applications.[3] You don't solve it by instructing the model to behave. You solve it at the tool layer.

- **Tool contracts as typed interfaces** rather than free-text descriptions. Name, parameters, scope, side-effect class (read, write, or destructive), and an explicit idempotency guarantee.
- **Authorisation against the human's real entitlements**, never against entitlements the model claims. The policy layer is the authority. The agent is a requester like any other.
- **Consequential writes route to a gate before execution**, never after. The agent proposes. A scoped approver disposes.
- **Every consequential action gets a defined rollback** or compensating action, designed like a transaction boundary rather than left to a postmortem.

Here's a useful test. Are the agent's permissions narrower than any single human it assists? If they're broader, you've quietly built a privilege-escalation path with a friendly interface.

06 · EVALUATION

The investment nobody makes on day one

Evaluation buys you more than anything else you can build early. It's also the thing most reliably deferred until after the first incident.

The test set

Build it from real historical cases. Stratify it deliberately to include edge cases, adversarial inputs, and the failure patterns your domain experts already know about. A sample of typical cases won't do. Practitioner guidance in 2026 lands on several hundred cases before an aggregate metric means anything.[4]

Judging an agent that acts

When a system takes actions rather than answering questions, the target shifts from "is the answer correct" to **"was the trajectory correct."** Right tool, right arguments, right order, correct recovery from a failed call, no unauthorised side effects. That takes trace-based evaluation against a known-good trajectory. Most retrieval-era tooling was never built for it.

Where LLM-as-judge breaks

Judges show length bias, favoring longer answers regardless of correctness. They show agreeableness bias, defaulting to approval rather than genuinely assessing. And they miss trajectory quality entirely when scored only on the final output.[5] Calibrate any judge against a human-labeled gold set before you trust it. Never let it be the only gate on a consequential release.

THE LOOP MOST TEAMS NEVER CLOSE

A new failure category found in production goes back into the offline test set. No prompt, tool, or model change ships without a green run against it. Without that return path, every incident surprises you twice.

07 · ACCOUNTABILITY

Who is answerable when the agent acts

This is the control that lets a risk function say yes. It’s architectural rather than procedural.

- **Name the owner before the workflow, not after the incident.** Every consequence class gets one named accountable human role. A write, a customer-facing action, a financial commitment. Not "the team." Not "compliance will catch it."
- **Approval friction scales with consequence.** Low-risk reversible actions proceed with logging. High-consequence or hard-to-reverse actions need approval before execution. Get this wrong in either direction and the project dies. Approve everything and you gain nothing. Approve nothing and the risk team blocks it entirely.
- **The audit trail is the mechanism rather than the paperwork.** What evidence was used, what was proposed, who approved it, and why. All reconstructable on demand.

> *A named human approving four hundred actions a day without engaging with any of them is accountability in name only.*

That last failure is real and worth saying plainly. Human-in-the-loop done badly gives you the appearance of control while removing the substance. What decides whether review is genuine or theatre is approval-queue design. What the reviewer sees, how much context arrives with it, and whether the friction matches what’s actually at stake.

08 · STATE OF THE ART

What changed in 2026, and what did not

Capability is improving quickly. METR's time-horizon benchmark measures the length of task, in human-expert time, that an agent completes at 50% success. It's been doubling roughly every four months since 2024, up from a seven-month doubling before that.[6] That's real. It's also a benchmark largely on coding tasks measured at 50% success, which is a very different thing from production reliability.

Tool interoperability consolidated. The Model Context Protocol is now the de facto standard across the major providers, and a mid-2026 spec update added centralized enterprise authentication.[7] Audit trails, SSO-integrated gateway auth, and configuration portability across environments are all mid-rollout rather than finished.

Frameworks consolidated too, around graph-based stateful orchestration with native protocol support, rather than ad hoc chaining.

Cost stopped being the constraint. Inference costs fell roughly fortyfold between 2023 and 2025.[8] What binds enterprise adoption now is reliability, evaluation, and governance. That's exactly why this paper is about those rather than model selection.

Still unsolved, and worth saying so

- Near-100% reliability on long-horizon tasks. The benchmarks measure 50% success, which isn't production-grade dependability.
- LLM-as-judge is still a heuristic. It needs calibration and human oversight rather than replacing either.
- Computer-use agents are not yet trustworthy unattended in regulated environments.
- Long-term agent memory has an unresolved staleness problem. Agents cite facts that were true when stored and have since changed, with no mechanism to notice.

Naming these openly isn't hedging. If you've already been disappointed once, it's the only thing that makes the rest of this credible.

09 · WHERE YOU ACTUALLY ARE

A readiness assessment worth scoring honestly

Ten questions, each scored zero to four. Zero means not present. Four means mature and enforced.

DIMENSION	THE QUESTION
Evaluation	Do you have a representative test set with known-good trajectories gating every release?
Grounding	Can you trace every agent action to a specific, timestamped, sourced fact, with conflicts handled explicitly?
Least privilege	Are the agent’s tool permissions narrower than any single human role it assists?
Accountability	Is a named human accountable for every consequence class the agent can trigger?
Observability	Can you reconstruct what the agent did, why, and with what inputs and outputs, after the fact?
Reversibility	Does every consequential action type have a defined rollback or compensating action?
Economics	Do you have a per-workflow cost and latency budget, actively monitored and alerted?
Access	Do the systems involved have proven, entitled API access at the granularity your workflow needs?
Topology	Did you evaluate a single-agent or deterministic alternative and rule it out explicitly?
Change control	Are prompt, tool, and model changes versioned, reviewed, and gated by the regression suite?

0–15 · DIAGNOSTIC 15–30 · PILOT-READY 30+ · SCALE

Below roughly fifteen out of forty, a production-shaped pilot isn’t credible yet and diagnostic work should come first. Between fifteen and thirty, a pilot is reasonable with named gaps closed alongside it. Above thirty, your live question is rollout readiness rather than first production.

Let the evidence decide the next step

The instinct after a list like that is to go build all of it. That’s the wrong conclusion. Score yourself honestly. Pick the workflow where the value is clearest and the access is real. Then let the evidence decide whether the next step is a diagnostic, a pilot, or a rollout. Not the ambition.

The most common thing we find in week two isn’t a technical obstacle. It’s that the assumed data access needs an entitlement nobody has requested, or that nobody is empowered to say what a good decision looks like. You want to hit either of those in a bounded diagnostic rather than month four of a build.

How Palmetto works on this

We design agentic systems around a real operating workflow rather than a generic assistant. And we scope the control model with the workflow rather than after it. Permissions, evaluation, approval points, and escalation all designed together.

Every engagement opens with a bounded diagnostic ending in a decision brief. When the honest reading is that you shouldn’t proceed, that’s what it says. Architecture decisions, tool contracts, evaluation assets, runbooks, and the operating backlog all go to your team. And your people stay accountable for the decisions that matter.

Palmetto Interactive palmettointeractive.com · support@palmettointeractive.com

REFERENCES & NOTES

1. Gartner, "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027," press release, 25 June 2025. A vendor-analyst prediction, not an observed outcome.
2. MIT NANDA, "The GenAI Divide: State of AI in Business 2025." Based on a review of 300+ public initiatives, interviews with 52 organizations, and a survey of 153 leaders. "Failure" here means no measurable P&L impact rather than system malfunction. The methodology has been publicly

disputed. Read the figure as directional, never as a measurement.

3. OWASP Top 10 for Large Language Model Applications. Prompt injection into an over-permissioned agent is the leading risk category.
4. Practitioner guidance on minimum evaluation-set size, 2026. Vendor engineering sources. Treat it as a rule of thumb rather than a validated threshold.
5. Documented LLM-as-judge biases (verbosity, agreeableness, and trajectory blindness) reported across evaluation-tooling engineering blogs, 2026.
6. METR, "Time Horizon 1.1," January 2026. Measures 50% success on predominantly coding tasks. Not a measure of enterprise workflow reliability.
7. Model Context Protocol specification update, July 2026, adding centralized enterprise authentication.
8. a16z, "Welcome to LLMflation." The multi-year trend is the citable claim. Precise current per-token figures are directional.

Where a figure here comes from vendor research, analyst prediction, or industry commentary rather than an audited study, we say so above. We deliberately left out the widely circulated agent-deployment statistics that carry no traceable source. Where a number materially affects a decision, we source it specifically during the engagement instead of relying on published estimates.