PROFORMA GLOBAL RESEARCH

Implementing Your First AI Agent Pilot

Matt Rollings, Founder and Principal, Proforma Global

Briefing · 2026-05-31

Executive Summary

Background. 95% of enterprise AI deployments produce no measurable return on investment. MIT put that number on it, and a rate that high means the design and deployment skills that lead to working systems are still rare. Agent design and deployment is new and mostly unexplored, and nearly all initiatives fail. Given the low success rate, it is highly likely that the projects that succeed meet minimum viable thresholds, not best practices. There are too few successes to realistically show what good really looks like. Most teams are finding the pitfalls the expensive way, in production. Vendors have filled the gap with confident roadmaps and big claims about cost reduction, few of which hold up against the real problems that inevitably arise during business as usual. A finance leader walking into these initiatives needs a way to tell a credible plan from a rehearsed one. The five questions below are where to start.

Approach. Ask a vendor five questions when it claims to know AI. Each one surfaces a core element of success and why it matters.

1. Five Questions to Ask a Vendor

1. What percentage of the tool should be driven by LLMs versus deterministic functionality?

An agent solution built mostly from LLM calls and tool calls, with little determinism, turns fragile and costly. LLMs are meant for judgment, decisions, and reasoning - not execution. Challenge every step to ensure it needs an LLM at all. Vendors keep deploying AI on top of broken processes when the simpler move fixes the process first so it can be automated.

A vendor who understands this gets specific about how little of the system needs the model and walks you through which steps run as fixed logic and which need judgment. Be wary of anyone who describes an agent that reasons its way through the whole job, or who cannot tell you what runs as plain code and what runs through the model.

2. How do we apply governance so that it is always enforced?

Governance exists only when something enforces it. An LLM cannot enforce governance. All prompts and LLM activities are merely hints toward outputs and behaviors. Governance has to run deterministically, through a traditional enforcement mechanism that fires before anything invokes the LLM. Read the implementation-level details here: Governance as Middleware.

Any and all governance that runs through a prompt or an LLM-driven output or tool call will fail a percentage of the time. Even a 99% success rate is not an acceptable pass rate for governance-related activities. Imagine an agent posting a journal entry without approval 1% of the time. Be wary of a vendor who says the model is trained or instructed to follow the rules. A model can always reason its way into ignoring them.

3. How do we keep AI deployments consistent as they grow?

A handful of general rules govern a successful rollout.

1. An agent that tries to do everything does nothing well. Every agent should have clearly defined scope.
2. Striving for elegance too early will create more problems than it solves.
3. Be willing to redesign entire agent frameworks as your architecture and processes evolve. Your first 12 months of deployments will be highly flawed.

The most consistent thing about your deployments will be the need for rapid change and adaptation as the internal competence and underlying architecture improves.

Ask how data quality may impact deployment strategy, and where the data quality issues typically reside.

4. What are the two or three use cases to start with, and why?

Start with simple processes that are largely deterministic and require light judgment. Tasks that follow a reliable workflow make the best candidates. They run roughly 90 percent deterministic and call an LLM only for the judgment step. The office of the CFO finds its most natural starting point in the month-end close, automating reconciliations across financial systems. The second and third use cases should add complexity slowly in task structure and logic, evolving the overarching design as they go.

5. How do we validate economic viability before committing to a full deployment?

Economic viability is a tricky area, as cost is driven by the number of calls to an LLM, and the volume of information an LLM needs to process. Validating the economics is purely a math formula, but assumptions are critical. Agreeing on concepts and guardrails on how and when LLMs can and will be used is a core premise that will protect long-term ROI.

Many solutions presented today rely on expensive and unreliable designs. More LLM calls in the process means more cost. An action that saves two hours of a person's time may cost ten dollars routed through an LLM, and the same action redesigned to run deterministically often costs pennies. Common promises of agent swarms and collaboration across multiple personas result in sky-high token consumption.

A credible answer comes with strategies to limit LLM calls and minimize the tokens sent for analysis. The result is a reasonably accurate cost per transaction at production volume and a clear account of how the vendor reached it.

---

2. Common Pitfalls of Agent Deployments

Most agent solutions fail in deployment because teams misunderstand and misrepresent the real issues. Organizations overestimate how well AI solves complex multi-domain problems. Context pollution is the cause. If you feed an LLM information from many areas at once, the outputs and outcomes deteriorate significantly. Generally, an approach that enforces one task per agent call is the rule we have found to be overwhelmingly successful. Account reconciliations do not need fifty agents, but they do need the work broken into atomic components so the agent does not get confused.

You quickly find that each step should be self-contained. Your team pulls in information from different systems as it moves through development, and it builds your new process into the agent's framework.

The implementation effort comes down to three things.

1. Assess the complexity of the problem to decide where an LLM call is needed to make a decision.
2. Recognize that the more freedom you give an LLM to make decisions, the more error-prone the process becomes.

3. Separate the process into its parts, the automation, the orchestration, the reasoning, and the intent.

Teams feel a built-in pull to build a universal problem solver, because people hope that is what AI agents can be. Agents are not there yet, and the reasons run architectural rather than temporary. Where Agent Orchestration Breaks makes the case against the universal orchestrator.

---

3. Future Scalability Pitfalls

Pilots exist to prove concepts and demonstrate functionality. An organization that attempts to futureproof an agent's architecture during the pilot creates long-term rework. Architectural stability and scalability belong in the follow-on projects once the initial solution is live. The data layer becomes a significant investment, and the organization should build it after it understands the problems LLMs introduce.

Avoid self-learning techniques early. Capture errors and process inconsistencies first. Self-Learning Architectures makes the case for holding self-learning until the foundation is in place, and The Agentic AI Maturity Curve in Enterprise Finance takes up which architecture suits which stage.

---

4. The Economics Have to Be Reasonable

A pilot is intended to prove the concept, and enable learning. A key point of differentiation is showing the economics are reasonable, not optimized. You are not trying to drive down cost yet. You are checking that the approach is not economically doomed before you commit to scaling it.

The thing to watch is the cost driver. In an agent system the number of LLM calls drives cost, not the number of tasks completed, and two designs that produce the same output can differ by many orders of magnitude in what they cost to run. A design that puts a model call in every step can look fine on a handful of pilot transactions and still be ruinous at volume. The pilot should give you enough of a read to know which kind you are holding.

A capable vendor can tell you roughly what a transaction costs and where that cost comes from. The real optimization is the deterministic redesign coming much later as individual tasks are scoped to in-house models where appropriate and the data architecture foundations are constructed.

---

5. Control, Audit, and Accountability

Finance requires more than success or failure of a given task or workflow. Finance demands that the agentic workflow be auditable and repeatable. Anything that updates or reports on financial information has to be right 100% of the time. A model carries a non-zero hallucination rate and cannot meet that bar on its own, which is why the financial data is handled by deterministic actions that keep the model out of the path that produces or changes a number.

Auditability of agentic solutions requires three things to be true.

1. Every action the agent takes has to be attributable and logged in a form an auditor will accept. Prompts and LLM responses do not pass a detailed audit on their own.
2. Any step that posts, approves, or changes a final number needs a human checkpoint and cannot be left to model judgment.
3. The controls must run deterministically and outside the model, for the same reason governance does, because the model reasons about whether to comply and a control cannot be probabilistic.

A pilot that produces correct numbers but cannot explain or evidence how it reached them has not removed risk. The pilot has only moved the risk behind an opaque wall. Evidence and control are part of the deliverable, not a later add-on.

---

6. Define Success Before You Start

Most enterprise AI pilots fail review because no one agreed in advance on what success was, not because the technology failed to run. Write down three things before the pilot begins. The baseline records what the task costs and how long it takes today. The target names the specific, measured improvement you are testing for. The kill criteria state the result that would tell you to stop.

A pilot without a baseline cannot be evaluated, only argued about. Naming success up front separates a pilot that informs a decision from a permanent demonstration nobody is willing to cancel.

7. When to Fund the Next Phase

A pilot earns the next phase once it satisfies three conditions. The economic model has to hold at production volume. The process has to be controllable and auditable to the standard finance requires. The team has to understand the work well enough to scale it. A pilot that leaves those unmet earns an extension, not a scale-up. Scaling an unproven design multiplies both its cost and its risk.

The architecture conversation becomes appropriate at this point and not before. It takes in the semantic layer, the data investment, and the move from one agent to many. A program compounds when it matches that investment to where the organization actually is, and it stalls when it chases where the organization hopes to be.

8. Closing

A well-scoped pilot stays small and quick to run, the cheapest way to learn what your organization is actually dealing with before you commit real budget. Everything in this paper helps you scope that pilot and spot a vendor who is improvising. It will not build the system for you. The building is where the 5 percent who deliver separate from the rest. They make the logic run as code, enforce governance before the model can act, build a shared data layer that holds up as you add use cases, and keep a record of every action an auditor will accept. We do that work, and we bring it to a pilot you are scoping.

DOWNLOAD PDF

Work with Proforma Global

The thinking in this paper is public; the methods that turn it into a working system are not. If it fits a problem your team is working on, that is what we bring to an engagement. Start a conversation: **info@proforma.global**.