PROFORMA GLOBAL RESEARCH

I Have Working AI Agents. What Do I Do Next?

Matt Rollings, Founder and Principal, Proforma Global

Whitepaper · 2026-06-02

Executive Summary

Background. Your initial rollout of AI pilots was successful. A handful of chatbots, knowledge bases, and agents are up and running, meeting their initial use cases. Now the task is to deploy agents across the organization at scale. Pilots do not need to follow best practices, meet economic targets, or enable auditability. Your organization's success now rests on a consistent, deployable framework that can be readily understood, replicated, and extended. The most challenging areas of the deployment lie ahead, with heavy emphasis on data architecture, orchestration, observability, and economics. Significant redesign will be required, and the learning curve to a new enterprise framework is steep.

Approach. This piece sets out how to go from one working agent to a consistent capability across functions, in the order that keeps it reliable and affordable. The governing discipline is the one we apply to every engagement: make it right, then make it fast. Get the architecture, the data, and the standard correct first, then optimize the economics. Poor economics are primarily addressed by technical design and process elegance, neither of which deploys quickly. Every organization pays tuition to reach this level. Aiming for perfection during each pilot's build to be functional and scalable on its own is the most expensive and least efficient method of learning how to deploy AI at scale.

1. Learn From a Few Dissimilar Pilots, Not One Win

The readiness to scale comes from what the pilots taught you. One successful pilot simply does not teach the organization enough about running AI at scale. A single working agent proves that one task, and its subsequent process, can be done. It does not tell you which elements are driving success. With only one example you cannot yet say what good looks like. Scaling on a single result is effectively pure risk since the probability of finding the critical design or workflow flaws is low.

The deliberate approach is to run two or three pilots chosen to be dissimilar: different types of use case at differing levels of complexity. Dissimilar is the operative word. Three variations on the same reconciliation rarely add additional learnings and value. A reconciliation, a forecast, and a customer-facing query each break in different ways and expose a different requirement. There is no single combination to test, because each pilot probes something the others do not:

1. how an LLM handles data-quality verification
2. whether it can reconcile across systems
3. what it costs to automate the monthly close

The number of agents and variety of use cases/designs matter. Two or three is enough to see what generalizes without spreading a young program too thin. If you proved a single use case and are now under pressure to scale, the first move is not to build the enterprise platform on that one result. It is to run a couple more, deliberately unlike the first, and prove the design principles work on different data sets and problem types before you commit the foundation to it. What that first pilot should establish is the subject of Implementing Your First AI Agent Pilot.

---

2. Now the Foundation Earns Its Place

During the pilot we advise against building the shared data layer and the formal definitions first. A pilot built on an untested, struggling architecture tests the foundation, not the agent. Significant risks emerge if decisions are made on an immature architecture and data layer, or on a fragile business process. That advice no longer applies once the pilot is behind you, provided that pilot correctly evaluated the agent itself.

The pilot taught you what the problem actually is, and the second and third use cases need exactly the foundation you were right to defer: the data layer, the semantic definitions, the governed feeds the agents draw on. Built now, each new use case reuses the last instead of starting from raw tables again. The investment that was premature before the pilot is the gating investment after it. Its requirements are the subject of the Data Architecture for Enterprise Agents series, and the semantic layer at its center is treated in Semantic Layers in Enterprise Agent Systems.

3. Make It Right, Then Make It Fast

Make it right, then make it fast is the engineering principle that governs most successful deployments with any technology. Correctness must come first: an agent that is fast and fragile, with no verification between it and the ledger, is worse than no agent. A wrong number causes significant pain downstream long after it is produced. Optimizing runtimes only matters once the execution of the task is repeatable and carries the trust of the team. Optimization comes second, and it is mostly an economics problem.

We do not hand a client a full roadmap of every optimization. The program is theirs to run, and the order depends on the organization. However, a small set of technical best practices separate agents that scale from agents that merely accumulate technical and functional process debt.

- **Standardize the data feeds.** Each agent should draw on one governed definition of each hierarchy and data point. Agents should not query raw tables and re-derive revenue, or margin, or headcount each in its own way. The moment two agents compute the same figure differently, the numbers stop reconciling, and the more agents that share the drift, the more expensive it is to unwind. One feed, one definition, many consumers.
- **Build the semantic layer.** Standardized feeds carry values while the semantic layer carries meaning. Building the semantic layer is arguably the most challenging aspect of the architecture as it requires synchronizing the way the agent can relate to the data and the business's dimensionality through the lens of business process and cycles. It carries each function's definitions, the metadata, and the relationships that let an agent reason about a business rather than manipulate its raw numbers. The semantic layer is what makes an agent reliable beyond the handful of tasks it was hand-built for.
- **Make the system promptable.** The reliability of an agent is largely the reliability of the context it is given. Assembling the right context for each task, deterministically and from a governed source, rather than stuffing a prompt and hoping, is what separates an agent that holds up from one that drifts. We treat the engineering of this in Dynamic Context Assembly.

These three practices are a representative sample, not the whole program. The point is the order: get the data and the definitions right first. A program that optimizes cost before they are right is paying to build the same thing twice. How the cost comes down, once the work is right, is where orchestration comes in.

4. Orchestration Is Where the Economics Are Decided

The largest technical lever on cost at scale is orchestration: What steps are performed, who performs them, and which models get assigned for which tasks. The organization should eventually identify how to leverage in-house models to drive orchestration where possible to reduce cost as it matures. Orchestration is purely a technical problem and it is the one that most directly sets the unit economics.

A model call is the expensive operation, and naive orchestration spends one on judgment where none is required. The main issue with enabling judgment during orchestration is that models will constantly defer and ask for more information, even if not required. If the organization is running one agent the waste is tolerable. Across a function it compounds and often is the difference between a capability that pays for itself and one that scuttles the initiative.

Good orchestration is the discipline of spending model calls only where judgment is required, while following a defined process for a particular use case in a repeatable fashion. The move is to reengineer the work so the deterministic majority runs as code, with the model reserved for the decisions only it can make. Context is assembled once and reused rather than rebuilt at each step. The judgment that remains runs on the cheapest model that can do it, kept in-house and on smaller models where the volume justifies the infrastructure, because routine work carried on frontier-model pricing is how the unit cost compounds. Reengineering the coordination between steps, as much as the steps themselves, is where the economics of scale are won. It is also a reliability concern, because an orchestration that routes a step to the wrong tool or the wrong agent is wrong before the model is ever asked.

5. One Framework Across the Functions

Scaling is not defined as ten agents running in production. Operating at scale is ten agents that share a foundation, design principles, and observability to understand the health of the entire framework. If a function is empowered to build its own agent, the result is a sprawl of incompatible systems that are individually defensible and collectively unmaintainable.

The same concept means different things in different parts of the business. Without a shared definition, each new agent attempts to rediscover what revenue or a closed period is. The end result is an unknown and oftentimes silent misinterpretation that may take months to be identified. The answer is a consistent framework: each agent scoped tightly to its function and bounded in what it owns, but drawing on the same data layer and built to one standard. Functions should own their processes and use cases, but the architecture should be globally enforced.

We do not advocate for agents to use advanced architecture unnecessarily, but they should be easily adapted to it as the need arises. Without it, extending agentic capability across the business becomes painful, inconsistent, and costly, which is the outcome this whole piece exists to prevent. Who owns that standard, and why it sits with a single accountable role rather than with each function, is taken up in Your Board Wants AI. Where Do You Start?.

---

6. Governance and Observability Become a Control Plane

In the pilot, governance could be a human checkpoint: one process, one reviewer, a person in the loop. Across many agents acting on many processes, that does not hold. The checkpoint has to become a control plane, enforced in the infrastructure rather than asked of each team.

Every action an agent takes should be attributable, logged to an audit standard, and constrained by rules that sit outside the model. A probabilistic reasoner cannot be trusted to enforce a deterministic rule on itself. That same record is the observability the program runs on at scale: when an agent produces a wrong result, the trace of what it did, on what data, under which rule, is what makes the fault findable instead of a mystery to reproduce. At one agent the logging looks like overhead. At ten it is the only thing keeping the capability auditable, and it is infrastructure to be built, not a per-pilot afterthought. Why enforcement belongs outside the model, in middleware rather than in the prompt, is argued in Governance as Middleware.

---

7. When, and Whether, to Let Agents Learn

The temptation at scale is to let the agents improve themselves. Hold it. Most of what an agent would learn from its own errors is a deferred fix for a broken process, not a genuine rule, and an agent that learns from a broken process learns the breakage.

Capture the errors and the inconsistencies first, fix the process, and only then consider letting a system adapt, under tight discipline and only once the foundation underneath it is solid. Self-improvement is a capability you earn, not one you switch on. The architecture for doing it safely is the subject of Self-Learning Architectures.

8. Match the Architecture to the Stage You Are At

The recurring trap at this point is building the architecture you aspire to instead of the one your current scale needs. A program running its second use case does not need the enterprise agent platform, and building it there burns months on infrastructure no current workload justifies. A program running ten agents by hand has outgrown the scripts that carried the pilot and pays for it in reliability.

Both are the same mistake, the architecture mismatched to the stage, in opposite directions. The right architecture is the one the current stage needs and the next one will not immediately invalidate. Knowing which is which, at each point on the curve, is exactly what The Agentic AI Maturity Curve in Enterprise Finance sets out.

9. Closing

Scaling agents is an architecture and governance problem, not a model problem. The model is rarely what stops you; the foundation, the standard, the controls, the orchestration, and the order you build them in are. Make it right, then make it fast: get the data, the definitions, and the framework correct and the economics follow; reverse the order and you optimize something you will rebuild.

Knowing the order is not the same as building what scale depends on. That build, the data layer, the standard, the governance, the orchestration, the reengineering that makes the unit economics work, is the work we do. For a team with working agents asking what comes next, that is where it begins.

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**.