Building Agentic AI Right.
Design Controls Before You Scale.

A practical guide for CEOs and COOs building agentic AI for scale, not just for the pilot.

Introduction

A regional insurance carrier spent 14 months building an agentic AI workflow for claims intake. The pilot results and technology worked. The workflow failed in production within six weeks.

Not because of the model. Because the design was never built for scale. The decision boundaries were vague. The work had been mapped, not re-engineered. The human review points were bottlenecks dressed up as governance. Nobody had asked what the workflow needed to look like when volume hit, exception rates climbed, and the edge cases the pilot never saw started arriving daily.

That is the pattern. And it is not a technology problem. It is a design problem.

Most agentic AI initiatives fail between pilot and production. The gap is not capability. It is the design work that was skipped before build started. The organizations that get agentic AI right the first time are not the ones with the best models. They are the ones that treated design as the hard part and build as the consequence of getting design right.

This guide gives you five checks to close that gap before it costs you.

One more thing that most agentic AI designs get wrong from the start. A single FTE's work is rarely replaced by a single agent. At enterprise scale, the right design is multiple specialist agents, each accountable for a defined part of the workflow, coordinated by an orchestration layer that manages handoffs, exceptions, and escalations. That is a fundamentally different design conversation from traditional automation, and it changes how you think about re-chunking the work, defining decision boundaries, and measuring outcomes. The five checks in this guide are written with that architecture in mind.

What This Guide Covers

- Check 1: Have you re-chunked the work, not just mapped it?
- Check 2: Have decision boundaries and human review been designed for scale?
- Check 3: Has every persona the agent interacts with been designed for?
- Check 4: Has exception handling been stress-tested against production reality?
- Check 5: Has the workflow been validated in a production-mirror environment?

What Makes Agentic AI Different From What Came Before

Traditional automation executes a defined sequence. A human designs the steps, the system follows them, and exceptions route back to a human queue. The accountability is clear because the human designed every step and volume is predictable.

Agentic AI is categorically different. An agent, or multiple agents make decisions across multiple systems, entities, and accountabilities without a human checkpoint at each step. It does not just follow a sequence. It executes conditions, resolves exceptions, and takes actions autonomously within boundaries set at the design time.

At low volume, a basic agentic workflow can look like it is working. Exception rates are manageable, edge cases are rare, and the human review queue stays thin. Scale changes all of that. Exception combinations multiply. Edge cases that never appeared in the pilot arrive daily. Human review points that seemed manageable become the bottleneck that breaks the outcome. The reality is that those flows were never designed for scale, humans not trained, and interventions not built in.

The organizations that scale agentic AI successfully are not the ones that build the fastest pilot. They are the ones that designed for production complexity before they wrote the first line of code.

Five Checks Before You Build

CHECK 1

Have you re-chunked the work, not just mapped it?

This is the check most organizations skip entirely, and it is the one that determines whether agentic AI delivers real value or automates the existing mess at scale.

Before agentic AI, work in operations-heavy environments was fragmented by constraint. Location boundaries. System access limitations. Maker-checker requirements. SSO restrictions. Shift timing. Cycle time buffers built around human handoffs. The result was work split across multiple people, queues, and systems, not because that was the best design, but because those were the constraints that existed at the time.

Agentic AI removes most of those constraints. What used to require three people in two locations across two shifts, with a maker-checker loop and a manual data re-entry step, can now be reconceived as a single agent-managed workflow. But only if you question the structure before you design the agent. What is the expected outcome is a good question to ask, rather than what is the current flow?

The check is not: have you mapped the current process? It is: have you questioned whether the current process boundaries still make sense at all? If your agentic AI workflow looks like your existing process with automation layered on top, you have not re-chunked the work. You have automated the fragmentation. At scale, that fragmentation does not disappear. It compounds.

> **IN PRACTICE**
>
> *A healthcare revenue cycle operation built an agentic AI workflow for prior authorization processing. The initial design mapped directly onto the existing process: separate queues for clinical review, coding validation, and payer submission, each handed off sequentially. When the workflow went live, throughput improved by 18% but the handoff delays between queues remained. Re-chunking the work, treating the agent as a single accountable actor across all three stages, eliminated the handoff queues entirely and reduced cycle time by 61%. The technology was the same, it was just redesigned right.*

5-Step Action List

1. List every handoff, location dependency, system access constraint, shift-based split, and maker-checker loop in the current workflow.

2. For each constraint, identify whether it exists because it is necessary or because it was a workaround for a human or system limitation that no longer applies in an agentic environment.
3. Map what the work chunk would look like if those constraints were removed entirely, with the agent as a single accountable actor across the full sequence.
4. Document the difference between the current fragmented process and the re-chunked agent workflow, and make that gap visible to both business and technology before build begins.
5. Define the key business outcome of the workflow upfront. Then evaluate every step in the design against one question: is this delivering to that outcome specifically.

Recommendation: *If your agentic AI design maps directly onto the existing process, stop. You are about to automate a structure that was built for human and system constraints that no longer apply. Re-chunk the work first. The value case will change significantly.*

CHECK 2

Have decision boundaries and human review been designed for scale, not just for the pilot?

An agentic AI workflow without explicit decision boundaries is not a workflow. It is an autonomous system operating without a design.

Decision boundary mapping means specifying three things for every decision point: what the agent decides autonomously, what it escalates to a human, and what it cannot touch. In a pilot, loose boundaries are manageable because volume is low and exceptions are rare. At scale, the same loose boundaries produce a human review queue that nobody can clear, autonomous decisions that compound in the wrong direction, and an operating model that has lost control of its own process.

Human-in-the-loop is where most designs fail at scale. Designed purely as a governance requirement, HITL becomes a bottleneck. Review queues that nobody monitors. Escalation paths that create delays. Notifications that arrive too late to be useful. The question is not whether to include human review. It is where human judgment adds real value, what that person needs to see and do, and how the workflow sustains throughput while they do it, including thresholds and rules to come into the queue.

Quality agents, whether human or AI-assisted, need to be positioned at the right trigger states with the right information and the right time window. Introduced in the wrong place, they are the mechanism that kills the outcome the workflow was built to deliver.

5-Step Action List

6. Map every decision point in the workflow and classify each: autonomous, escalate to human, or out of scope for the agent entirely.

7. For each escalation point, design the human review experience: what the reviewer sees, what context they have, what action they take, and the maximum time the workflow can hold before proceeding. Bake capacity for this.
8. Stress-test the boundary design at projected production volume, not pilot volume. Ask what happens when the escalation queue is full.
9. Position human review and quality check points at the trigger states where human judgment adds genuine value, not at the points where governance requires a signature.
10. Validate the boundary design with both the operational team who will manage escalations and the business owner who is accountable for outcomes.

Recommendation: *If your decision boundaries and human review design were built for the pilot and not tested at scale, they will fail in production. The boundary map is not a governance document. It is the operating design of the workflow. Get it right before build starts.*

CHECK 3

Has every persona the agent interacts with been designed for?

Agentic AI touches more personas than traditional automation. The end customer whose application or claim the agent processes. The internal operator who monitors and overrides. The supervisor who is accountable for outcomes. Each has different trigger states, different information needs, and a different definition of what good looks like.

Designing for the agent without designing for each persona it touches is the same mistake as building a product without user research. In a pilot, workarounds are easy. At scale, the gaps in persona design surface as adoption failures, operational workarounds, and outcomes that do not match what the workflow was built to deliver.

Persona design at this level is not a UX exercise. It is an operational design requirement. The notification a supervisor receives when an agent escalates needs to contain exactly the right information for a decision to be made in the time available. The override interface an operator uses needs to be fast enough that it does not become the reason throughput stalls. Every person the agent interacts with needs to have been designed for before build starts.

IN PRACTICE

A life insurance carrier built an agentic AI workflow for underwriting support. The agent design was thorough. The underwriter experience was not. When the workflow went live, underwriters found that the agent's escalation notifications contained data summaries but not the specific risk flags they needed to make a decision quickly. The result was that underwriters were opening four to six additional systems to retrieve context the notification should have contained. Average review time per case was 23 minutes, against a target of 8. The agent was performing correctly. The persona design had not been done.

5-Step Action List

11. List every persona who interacts with or is affected by the agent: end customers, internal operators, supervisors, and any downstream teams who act on agent outputs.
12. For each persona, define the trigger states that require them to act, be notified, or have visibility into what the agent has done.
13. Design the notification and interface for each persona: what they see, when they see it, in what format, and what action they are expected to take.
14. Validate the persona designs with at least two real people in each role performing real tasks in a simulated environment, not a walkthrough.
15. Confirm that the persona and notification designs support the throughput and cycle time targets the workflow was built to deliver, not just the governance requirement.

Recommendation: *If the personas who interact with the agent have not been explicitly designed for, the workflow will underperform in production regardless of how well the agent performs in isolation. Persona design is not optional. It is the layer that determines whether the technology delivers the operational outcome.*

CHECK 4

Has exception handling been stress-tested against production reality?

Exception handling in agentic AI is categorically more complex than in traditional automation. A traditional system encounters a condition outside its design and stops. An agent may encounter combinations of conditions that were never explicitly modeled and will attempt to resolve them autonomously unless the design specifies otherwise.

In a pilot, exception rates are low and the combinations are predictable. Production is not a pilot. At scale, exception combinations multiply in ways that no design team fully anticipates. The question is not whether your exception handling covers the scenarios you designed for. It is whether it covers the scenarios you did not.

Graceful degradation must be designed, not assumed. When the agent encounters a condition outside its decision boundaries, what exactly does it do? Who does it notify? What does it log? How does the workflow recover? If those questions do not have specific answers in the design, the exception handling has not been designed. It has been deferred.

IN PRACTICE

A BPO operation running a claims processing workflow for a healthcare payer deployed agentic AI across a high-volume queue. Exception handling had been designed for the twelve most common claim exception types from historical data. In production, a regulatory change introduced a new denial reason code that triggered a combination of conditions the agent had not been designed for. The agent attempted autonomous resolution, produced incorrect outputs across 4,200 claims over three days before the pattern was identified. A designed exception path with a graceful degradation rule would have flagged the unknown condition on the first occurrence and routed to human review. The remediation cost was significantly higher than the design sprint would have been.

5-Step Action List

16. List the top ten exception scenarios from production data, not pilot data, and confirm each has an explicit handling path in the workflow design.
17. Run a structured edge case injection session with real operational staff to surface the scenarios the design team did not anticipate.
18. For each exception path, define the agent's graceful degradation behavior: what it does, who it notifies, and how the workflow recovers.
19. Design the unknown condition handler: what the agent does when it encounters a combination of conditions that falls outside every designed exception path.
20. Document every exception path and its handling logic as a formal part of the design specification, not a footnote in the technical build notes.

Recommendation: *If your exception handling was designed for the scenarios you anticipated and not stress-tested against production reality, the first unanticipated exception at scale will surface the gap. Design the unknown condition handler before build starts. It is the most important exception path in the workflow.*

CHECK 5

Has the workflow been validated in a production-mirror environment, not just a pilot?

Pilots run in simplified environments. Production is not simplified. It has data quality variation, legacy system latency, concurrent workflow conflicts, peak volume patterns, and user behaviors that no pilot ever fully replicates.

Validation means running the workflow end-to-end in an environment that mirrors production complexity, with deliberate injection of edge cases, exception conditions, and high-volume scenarios. It also means validating every persona touchpoint with real people performing real tasks under realistic conditions, not a scripted walkthrough.

The gap between pilot performance and production performance is almost always a design gap, not a technology gap. What surfaces in a proper production-mirror validation is almost always different from what the build team expected. That difference is precisely why the validation exists. Finding it before go-live costs a fraction of finding it after.

5-Step Action List

21. Build a validation environment that mirrors production complexity: real data patterns, legacy system connections, realistic volume, and peak load scenarios.
22. Run the full workflow end-to-end with deliberate injection of the exception scenarios identified in Check 4, including the unknown condition handler.

23. Validate every persona touchpoint with real people from each persona group performing real tasks under realistic time pressure, not a demo environment.
24. Run the validation at projected peak volume, not average volume. The design needs to hold at the top of the range, not the middle.
25. Do not proceed to production deployment until the validation has been completed, gaps have been addressed, and the design has been confirmed against production-level complexity.

Recommendation: *A workflow that has not been validated in a production-mirror environment has not been validated. Pilot success is not production readiness. Validation means testing the full complexity of production before it goes live, not learning what that complexity looks like after the first failure.*

The Bottom Line

Agentic AI is not a technology deployment. It is an operating model change. The organizations that get it right the first time are not the ones that move the fastest. They are the ones that treat design as the hard work and build as the outcome of getting design right.

The five checks in this guide are not a framework. They are the questions that the planning process consistently skips because the answers require slowing down before the build momentum has built. The organizations that ask them before build starts move significantly faster in production than the ones that discover the answers mid-delivery.

After 25 years running transformation across global enterprises in banking, healthcare, and operations, the pattern is consistent. The initiatives that stall at scale almost always had a design gap that was visible before build started. The discipline to surface it before the first line of code is written is what separates programmes that scale from programmes that get rebuilt.

If you read these five checks and found gaps in your current design, those gaps are not a reason to stop. They are a reason to design before you build.

If your agentic AI initiative needs a design review before build begins, a 15-minute discovery call is the right starting point. vervelength.com

Shalini Verma

Founder & Fractional Chief Transformation Officer, Vervelength

Co-author, The Enterprise Mindset: Playbook to Build and Scale AI-Enabled Solutions

connect@vervelength.com · vervelength.com · linkedin.com/in/shalini-verma-sv