

BEYOND MODEL CAPABILITY

# The Autonomy Gap: Why Agentic Workflows Stall Between Demonstration and Dependable Operation

A Practitioner's Perspective on Autonomous Decision-Making in the Enterprise

Giovanni Leonardi

January 2024



# Contents

|                                                                    |          |
|--------------------------------------------------------------------|----------|
| <b>1. Executive Summary</b>                                        | <b>3</b> |
| <b>2. The Shift From Capability to Consequence</b>                 | <b>4</b> |
| <b>3. Why the Demonstration Succeeds and the Deployment Stalls</b> | <b>4</b> |
| <b>4. Autonomy Is a Property of the System, Not the Model</b>      | <b>5</b> |
| <b>5. A Graduated Model of Autonomy</b>                            | <b>6</b> |
| <b>6. Building the Operating Envelope Before the Agent Acts</b>    | <b>7</b> |
| <b>7. What a More Effective Response Looks Like</b>                | <b>7</b> |
| <b>8. Conclusion</b>                                               | <b>8</b> |

# The Autonomy Gap: Why Agentic Workflows Stall Between Demonstration and Dependable Operation

Transformation · No. 363

Originally written January 2024 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

---

*“Autonomy is not a property you buy with a more capable model; it is a property of the system you build around it.”*

---

## 1. Executive Summary

---

Over the past year the conversation about autonomous systems in the enterprise has quietly changed its subject. It began as a question about capability — whether a model could plan, call a tool, read the result and decide what to do next. It has become a question about consequence — whether an organisation can let it. The distinction matters more than the enthusiasm of the moment suggests, because the capability that makes an agent interesting is precisely the capability that makes it difficult to govern.

The pattern I have observed across organisations experimenting with agentic workflows is remarkably consistent: the demonstration succeeds and the deployment stalls. A well-constructed prototype handles a curated task convincingly. The same construct, placed in the flow of real work, meets the conditions no demonstration contains — ambiguous inputs, missing data, unhandled edge cases, and the quiet, immovable expectation that someone remains accountable for the outcome.

This paper argues that the binding constraint on autonomous decision-making is not the capability of the model but the readiness of the enterprise to delegate. Autonomy is not a property you buy with a more capable model; it is a property of the system you build around it — the boundaries you set, the fallbacks you engineer, the decisions you are willing to make reversible, and the accountability you are prepared to hold. Organisations that treat autonomy as a model feature consistently over-deploy and then retreat. Organisations that treat it as an operating decision — granted deliberately, one decision class at a time — make slower but durable progress.

The recommendation that follows is a graduated model of autonomy. Classify decisions by their reversibility and their consequence. Grant autonomy only where a wrong decision is bounded and recoverable. And build the operating envelope — the data, the guardrails, the escalation paths and the audit trail — before the agent is permitted to act, rather than assembling it in the wreckage after the agent fails.

*The question that decides whether an agentic workflow reaches production is rarely can the model do this? It is almost always what happens when it is wrong, and who answers for it?*

## 2. The Shift From Capability to Consequence

The last eighteen months delivered a genuine change in what software can do. Systems built on large language models can now interpret an instruction expressed in ordinary language, decompose it into steps, invoke tools to gather information or take action, observe what came back, and adjust. The early autonomous-agent experiments that circulated last year — the ones that set a goal loose and let a model pursue it through a loop of reasoning and action — were rough, often unreliable, and frequently absurd. But they demonstrated something that was not obvious before: that the reasoning-and-acting loop could be closed by a model rather than by a programmer writing explicit control flow.

That is the capability. The trouble is that most enterprise conversations have stopped there, as though closing the loop were the hard part. It is not. The hard part is everything that a closed loop implies once it is pointed at work that matters. A traditional application is deterministic in the ways that count: given the same input it produces the same output, its failure modes are enumerable, and when it does something wrong you can trace exactly why. An agentic workflow is none of these things by default. It is probabilistic, its failure modes are open-ended, and its reasoning — however fluent the explanation it offers — is not a reliable account of why it acted.

This is why the centre of gravity has moved from capability to consequence. When software merely advises, a wrong answer is a suggestion a human can discard. When software acts — sends the message, updates the record, approves the request, moves the money — a wrong answer is an event in the world that someone must now clean up. The step from advice to action is not incremental. It is the step across which all the difficulty lives.

## 3. Why the Demonstration Succeeds and the Deployment Stalls

The demonstration is seductive precisely because it is curated. Someone who understands the task has chosen a representative case, ensured the necessary data is present and clean, and framed the instruction with unconscious precision. Under those conditions the agent performs, and the room concludes that the capability is proven and the rest is engineering.

What the demonstration omits is the distribution of real work. In production the inputs arrive malformed, the data the agent needs is stale or missing, two systems disagree about the same fact, and the instruction is ambiguous in ways the author never noticed because a human colleague would simply have asked. The pattern that recurs is not that the agent fails catastrophically — it is that it fails plausibly. It produces an answer that is confident, well-formed, and wrong, and does so without any of the signals a human

reviewer uses to sense that something is off.

Three structural forces turn a successful demonstration into a stalled deployment.

- The long tail is where the work actually is. The curated case represents the centre of the distribution. The value — and the risk — sits in the tail: the exceptions, the unusual combinations, the cases that a rule-based system would have escalated. Agents are often strongest on the cases you least needed help with and weakest on the ones you did.
- Confidence is uncorrelated with correctness. These systems express uncertainty poorly. A fluent, decisive answer and a fabricated one look identical on the surface, which means a human overseer cannot triage by tone. Oversight that depends on catching the obviously-wrong answer will not catch the plausibly-wrong one.
- Accountability does not dissolve because a machine acted. When an autonomous step produces a bad outcome, the organisation does not accept "the agent decided" as an answer. The accountability rests, as it always did, with a person and a function. Until that person can see what the agent will and will not do, and can trust the boundary, they will not sign the deployment — and they are right not to.

The stall, in other words, is not a failure of nerve or a lack of engineering skill. It is a rational response to a system whose behaviour has not yet been made legible and bounded.

## 4. Autonomy Is a Property of the System, Not the Model

The most consequential misconception in the current discourse is that autonomy is a dial on the model — that a more capable model is a more autonomous one, and that waiting for the next generation is a strategy. It is not. A more capable model raises the ceiling of what an agent can attempt; it does nothing on its own to make the attempt safe, bounded or accountable. Those properties are engineered into the system that surrounds the model, or they are absent.

Consider what actually determines whether an agent can be trusted to act. It is not the model's benchmark score. It is whether the action it can take is reversible. It is whether the data it reasons over is current and correct. It is whether a boundary exists that the agent physically cannot cross — a spending limit it cannot exceed, a category of record it cannot touch, a class of decision it must hand back. It is whether a wrong action leaves a trail that lets you detect it, understand it and undo it. Not one of these is a property of the model. Every one of them is a property of the enclosure the organisation builds.

*A capable model inside a weak enclosure is more dangerous, not less. Capability without boundary simply means the system can go further before anyone notices it has gone wrong.*

This reframing is liberating in practice, because it moves the problem from a domain the enterprise cannot control — the pace of model improvement — into one it can: the design of decision rights, guardrails and fallbacks. It also explains why some organisations are quietly making progress with today's imperfect models while others wait for a capability

that would not, by itself, solve their problem. The former are building enclosures. The latter are waiting for a dial that does not exist.

## 5. A Graduated Model of Autonomy

If autonomy is a system property granted deliberately, then the central design act is deciding which decisions to delegate and how far. The instinct to answer this by domain — "we will automate procurement" — is the wrong cut. The right cut is by the character of the individual decision, along two axes: how reversible a wrong decision is, and how large its consequence.

These two axes produce a simple map. Where a decision is both reversible and low in consequence, autonomy is close to free — let the agent act, and catch the rare error cheaply after the fact. Where a decision is irreversible and high in consequence, autonomy should be withheld regardless of how capable the model appears — the agent may prepare, recommend and assemble, but a human commits. The interesting territory is the two middle quadrants, where the answer is not autonomy or its absence but a specific pattern of human involvement.

| Decision character              | Appropriate autonomy                                  | Human's role                         |
|---------------------------------|-------------------------------------------------------|--------------------------------------|
| Reversible · low consequence    | Act autonomously; sample for quality                  | Audits a sample after the fact       |
| Reversible · high consequence   | Act, but with automatic checkpoints and easy rollback | Reviews exceptions; can undo quickly |
| Irreversible · low consequence  | Act within hard limits it cannot exceed               | Sets the limits; spot-checks         |
| Irreversible · high consequence | Prepare and recommend only                            | Makes and owns the decision          |

The value of this map is that it turns an anxious, all-or-nothing debate into a series of bounded, defensible choices. It also directs engineering effort where it pays. Making a decision reversible — adding a rollback, a checkpoint, a hold period before an action becomes final — often does more to unlock safe autonomy than any improvement to the model, because it moves a decision from a quadrant where autonomy is forbidden into one where it is affordable. Reversibility is not a constraint on autonomy. It is the mechanism that earns it.

1. Enumerate the decisions inside the workflow, not the workflow as a whole. A single process usually contains decisions from every quadrant.
2. Place each decision on the two axes honestly, using the cost of the worst plausible wrong action, not the average case.
3. Grant autonomy quadrant by quadrant, and invest engineering in moving high-value decisions leftward — toward reversibility — rather than waiting for the model to earn trust it cannot structurally provide.

## 6. Building the Operating Envelope Before the Agent Acts

---

A graduated model tells you what to delegate. The operating envelope is how you make the delegation safe. In every durable deployment I have observed, the envelope was built before the agent was allowed to act — not retrofitted after an incident forced the issue. It has five components, and none of them is optional.

1. A bounded action space. The agent should be physically incapable of taking actions outside its remit. This is enforced not by instruction — instructions can be misinterpreted or overridden by a cleverly malformed input — but by the tools and permissions it is given. If the agent must never exceed a threshold, the threshold lives in the tool, not the prompt.
2. A trustworthy data foundation. An agent reasons over whatever data it can reach. Where that data is stale, contradictory or incomplete, fluent reasoning produces confident error. The unglamorous work of data quality, freshness and lineage is not a prerequisite the enthusiasm can skip; it is the ground the whole capability stands on.
3. Explicit escalation paths. The agent must know — and must be able to recognise — when a decision belongs to a human, and handing back must be a designed, low-friction path rather than a failure state. A system that can only succeed or fail, with no third option of "escalate", will paper over its uncertainty rather than surface it.
4. An immutable audit trail. Every action, and the reasoning and inputs that led to it, must be recorded in a form that survives for later inspection. This is what makes a wrong action detectable, explicable and — where the decision was reversible — undoable. Without it, the organisation is trusting a system it cannot examine.
5. A defined accountable owner. For every class of decision the agent makes, a named person and function must own the outcome. Autonomy does not move accountability from the human to the machine; it moves the human's work from making each decision to designing, bounding and monitoring the system that makes them.

---

*"Autonomy does not remove the human from the loop.  
It changes what the human in the loop is responsible  
for — from deciding each case to governing the  
machine that decides."*

---

The envelope is deliberately unglamorous. It is data hygiene, permissions engineering, logging and organisational design — the same disciplines that have always separated software that survives contact with production from software that demos well and dies quietly. The novelty of the agent does not exempt it from these disciplines. It raises the stakes on getting them right.

## 7. What a More Effective Response Looks Like

---

Drawing the threads together, the organisations making real progress share a recognisable posture, and it is almost the opposite of the prevailing one.

They start from consequence, not capability. Before asking whether a model can perform a task, they ask what a wrong performance would cost and whether it could be undone — and they let that answer, not the demo, decide where to begin. They begin, deliberately, in the reversible and low-consequence quadrant, where autonomy is cheap and the organisation can build the operational muscle — the monitoring, the auditing, the escalation habits — that harder quadrants will demand.

They invest in the enclosure rather than waiting on the model. Recognising that autonomy is a system property, they put their effort into reversibility, boundaries, data quality and audit — the things within their control — and treat each increment of model capability as a bonus that widens the envelope rather than a saviour that replaces it.

They keep accountability human and explicit. Rather than allowing "the agent decided" to become an ambient excuse, they name the owner for every class of delegated decision and give that owner the visibility and the controls to discharge the responsibility. This is not a brake on adoption; it is the thing that makes adoption defensible to the people who must answer for it, and therefore the thing that lets it scale.

And they measure the right failure. They do not celebrate the demonstration's success; they instrument the deployment's plausible-but-wrong actions — the confident errors — because that is the failure mode that determines whether an autonomous workflow is an asset or a liability. A deployment that cannot surface its own quiet mistakes is not ready, however well it performs on the cases anyone thought to test.

## 8. Conclusion

The capability is real, and it is not going away. Software that can reason, act and adapt is a genuine addition to what the enterprise can build, and the organisations that learn to wield it will hold a real advantage. But the advantage will not go to whoever adopts the most capable model soonest. It will go to whoever learns fastest how to delegate safely — how to decide, decision by decision, what to hand over and how far, and how to build the enclosure that makes the handover accountable.

Autonomy, in the end, is not something a model grants an organisation. It is something an organisation grants itself, deliberately and in gradations, by doing the patient work of making its decisions legible, its actions reversible, and its accountability explicit. The model closed the loop. Whether the loop is safe to close in production is, and will remain, a decision the enterprise has to make — and to own.