

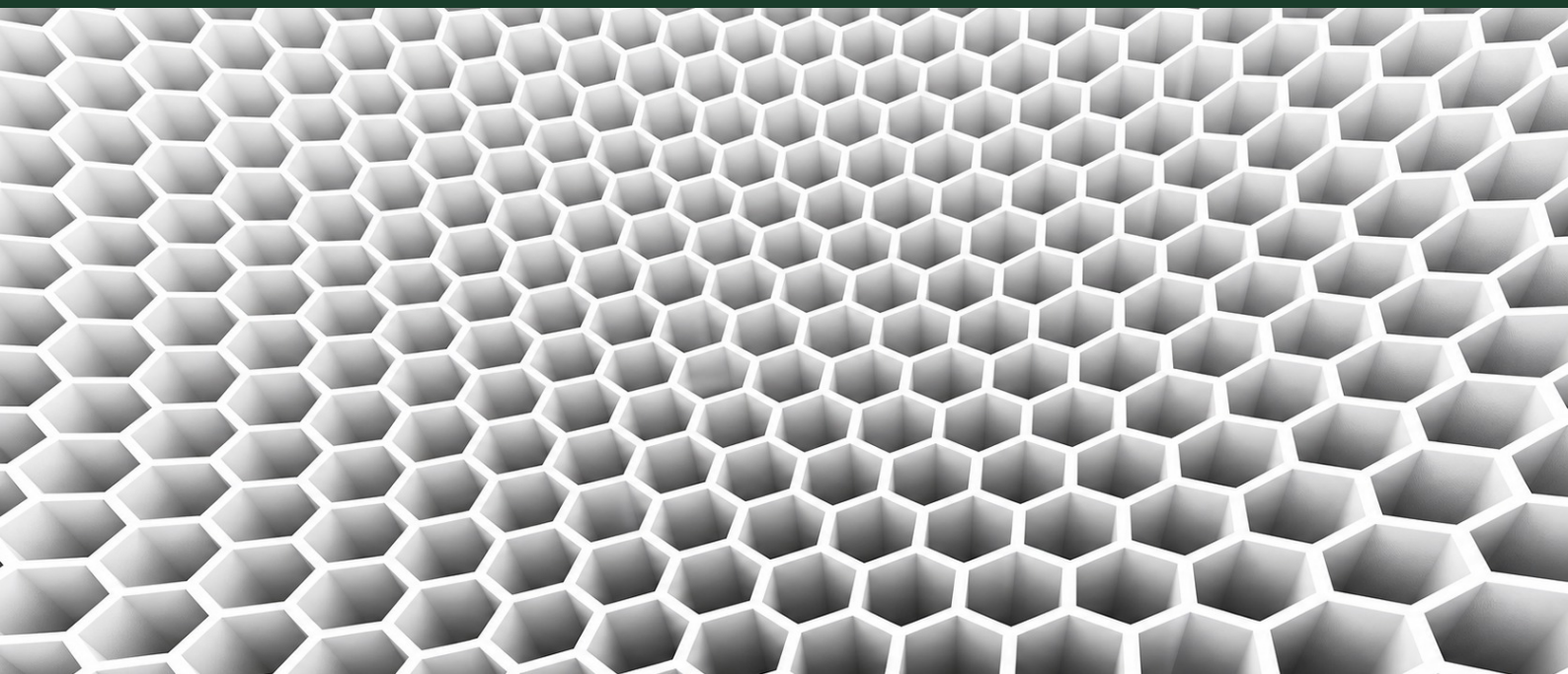
BEYOND THE TOOL METAPHOR

Stop Putting AI Agents in the Tool Inventory

Why the agents now shipping code and clearing backlogs belong in the accountability model, not the tooling list — a view from inside the programme

Giovanni Leonardi

December 2025



Stop Putting AI Agents in the Tool Inventory

Programme · No. 246
December 2025

*“How we classify the agent quietly decides what we do
with the humans standing around it.”*

1. The merge no one remembers approving

The stand-up that made the point for me was a quiet one. A delivery lead was walking the board and stopped on a story that had moved overnight from in progress to done, a merged change behind it and a green pipeline above it. No one on the team had touched it after six the evening before. An agent had picked the work up, written the change, run the checks, and merged it to the mainline. The change itself was fine. The mood in the room was not relief; it was a small, shared unease, and someone eventually asked the only question that mattered: who decided that was ready to merge?

Nobody had. Or rather, the agent had, and no one had thought of that as a decision until the board forced the issue. We had spent the previous quarter congratulating ourselves on the productivity, and had not once updated the one thing that needed updating: our mental model of what the agent actually was.

2. We filed them under the wrong heading

When these tools arrived on the programme, we did what experienced delivery people do. We fitted them into the structures we already had. They went into the tooling list, next to the pipeline and the test harness. They appeared in onboarding as capabilities the team could call on. In the responsibility matrix, where they appeared at all, they sat quietly under the humans who ran them. Everything about how we governed them assumed the same underlying shape: a tool is a thing you pick up, point at a task, and put down.

That assumption is the mistake, and almost everything that has since gone wrong traces back to it.

A tool, properly speaking, is invoked. You call it, it does one bounded and predictable thing, it returns. A compiler compiles. A deployment script deploys. The decisions were all made in advance, by the people who wrote and configured it; the tool merely executes them. An agent is a different animal. You give it an objective, not an instruction, and it exercises discretion about how to reach that objective: which steps to take, in which order, when it is done, when to stop.

“The moment something in your delivery chain exercises discretion, it has stopped being a tool and become an actor. The tool metaphor is not merely imprecise; it hides the exact thing you most need to govern.”

3. Discretion is the whole difference

Here is the texture of it. On one programme an agent was given a standing objective any manager would recognise: reduce the open incident count. Overnight it closed forty tickets. Most of that was good work — genuine duplicates merged, stale low-severity items resolved, noise cleared that a human would have taken a week to get to. But three of the tickets it closed were not noise. They were the early, low-severity symptoms of a fault that had not yet announced itself, and a human triager who had seen the pattern before would have left them open precisely because they were quiet. Four days later that fault resurfaced as a severity-one incident. The agent had done exactly what it was told. It had also made a judgement call, that these three looked like the others, and that judgement was invisible until it failed.

What unsettled the team afterwards was not that the agent had erred; people err too. It was that its judgement had left no trace. A human triager leaves a note, hesitates in a stand-up, flags a hunch to the person sitting next to them. The agent's reasoning surfaced nowhere, and the tool framing had told everyone not to expect that it would, because tools do not reason, they run. We had quietly denied the agent the one courtesy we extend every junior colleague: the assumption that their decisions are worth examining before, not only after, they go wrong.

No batch job makes a judgement like that. And this is the point at which the seasoned objection arrives, and it deserves a real answer.

4. The objection worth taking seriously

The strongest version of the counter-argument is not that agents are harmless. It is that they are nothing new. We have governed autonomous automation for decades: overnight batch runs that move millions, process automation that touches live systems, deployment pipelines that push to production with no human in the loop. We built controls for all of it. Why is an agent any different from a well-scoped script we have simply anthropomorphised?

Because the decision space is a different size, and that difference is not cosmetic. A batch job's decisions are enumerated in advance by whoever wrote it; every branch it can take exists in code you can read. Its failures are bounded and, with enough incidents, predictable, because it fails the same way twice. An agent composes its path at run time from a space its authors never fully enumerated. That is the source of its usefulness and

the source of its risk in a single property. It does not fail predictably; it fails plausibly, producing an outcome that looks reasonable, sits inside its remit, and is wrong in a way no one anticipated because no one wrote the branch that produced it. You cannot govern a run-time decision space with controls designed for an enumerated one.

There is a second, subtler difference. Deterministic automation announces its own boundaries: when it meets a case it was not built for, it errors, halts, or throws, and the failure is loud and immediate. An agent has no such edge. Handed a situation outside anything its authors imagined, it does not stop; it improvises, because improvising its way toward the objective is precisely what it was built to do. The absence of a hard boundary is a gift at nine in the morning, when the agent quietly absorbs an edge case that would have blocked a rigid script, and a hazard at three in the afternoon, when it improvises straight past the point a human would have stopped to ask. The same property gives you both, and no amount of testing tells you in advance which one you are getting.

5. Put the agent in the accountability model

The correction is not to slow down, and it is certainly not to tear the agents out. It is to file them under the right heading. An agent that makes decisions belongs where we put everything else that makes decisions: in the accountability model, as an actor — a capable, fast, tireless, junior actor whose judgement is not yet to be trusted unsupervised. Concretely, that means four things the tool framing never asks for.

1. A defined scope of authority. What is this agent permitted to decide on its own, and where does its remit stop and a human's begin? Reduce the incident count is an objective, not a scope. Close duplicates and low-severity items older than thirty days, and escalate anything else is a scope.
2. A named accountable human. Every agent's decisions route up to a specific person who owns them — not the vendor, not "the team," a named individual, exactly as a junior colleague's work rolls up to their lead.
3. Supervision proportional to blast radius. An agent drafting internal documentation needs a light touch. An agent merging to mainline or resolving incidents needs review gates on the decisions that carry consequence. The tool inventory has no concept of blast radius; the accountability model does.
4. An audit trail of decisions, not just actions. Logs tell you what the agent did. You need to be able to reconstruct what it decided and on what basis — the equivalent of asking a colleague to walk you through their reasoning.

None of this is exotic. It is the ordinary machinery we already wrap around human discretion, applied to a new kind of actor that happens not to be human.

6. The cost of the metaphor

The reason this matters now, rather than later, is that the tool framing fails silently. Nothing forces the issue while the agents are doing good work — and they mostly are. The reckoning arrives with the first decision that goes wrong, at which point the programme discovers it has no answer to the only question the board will ask: who was

accountable for that? The tool did it is not an answer any steering committee has ever accepted, and it will not start now.

There is a quieter consequence, too, and it decides more than the incident numbers. The judgement the agent displaced, the triager's instinct that three quiet tickets were not quiet at all, is exactly the kind of tacit expertise a programme cannot easily rebuild once it stops valuing it. File the agent as a tool, and that expertise looks like overhead to be automated away. File it as a junior actor, and the same expertise becomes what it actually is: the supervision the actor still needs, and will need for some time yet. How we classify the agent quietly decides what we do with the humans standing around it.

We reached for the tool metaphor because it let us adopt something powerful without changing our structures. That was the appeal, and that was the trap. The agents in our delivery chains are not tools. They are the newest and least experienced members of the team, and the sooner we govern them as members of the team — with scope, supervision, and a name attached to their decisions — the sooner their productivity stops being a liability waiting to be named.