

Agent harnesses: why human-in-the-loop is the 2026 architecture, not a compliance checkbox

Most conversations about AI agents in the enterprise are still framed as a trust problem: how much autonomy do we dare grant, and how do we prove to a regulator or a board that we didn't grant too much. That framing is backwards. The evidence from the last year of production deployments points somewhere else entirely — towards architecture, not permission.

The problem: agents drift, and the re-explaining tax

Start with the gap between what agents can do and what organisations actually let them do. Stanford HAI's 2026 AI Index reports that agent success on the OSWorld computer-use benchmark rose from roughly 12% to about 66%, and real-world task success reached 77.3% [10]. Meanwhile, McKinsey's 2026 survey work finds that only 23% of organisations are scaling agentic AI anywhere in the enterprise, and no single business function has pushed scaled adoption past roughly 10% [9]. Capability is running well ahead of production use.

Talk to the CTOs actually running agents in production and the stalling point usually isn't a dramatic failure. It's smaller and more corrosive than that: agents drift. A task gets picked back up — by the same agent an hour later, by a different agent, by a human reviewer — and someone has to reconstruct what happened and why before they can trust the next step. Call it the re-explaining tax: every handoff that loses context has to be paid for again, in human attention, before work can resume.

The fix isn't "more autonomy" and it isn't "less autonomy" either — both are still answers to the wrong question. The fix is a **harness**: the engineered layer that sits around a model and gives an agent a goal to pursue, rules it can't cross, gates where it must stop, and a log of what happened. Get the harness right and oversight compounds — each decision makes the next one cheaper to make. Get it wrong, and you're paying the re-explaining tax on every cycle, no matter how capable the underlying model is.

What a harness actually is

Harness engineering is now a distinct discipline, not an implementation detail bolted onto a model. A recent review argues that agent capability increasingly comes from *externalising* state, procedure and interaction structure — into memory, reusable skills, protocols, and the harness that coordinates them — rather than from further model training [3]. The harness is the unification layer: it doesn't make the model smarter, it makes the system around the model governable.

Four working parts do the job:

- **Goal** — what the agent is actually trying to accomplish, and for whom.
- **Rules** — the constraints and quality bar it may not violate, independent of what the model “wants” to do.
- **Gates** — the specific points where the agent must stop and get a decision before continuing.
- **Log** — the durable record of what happened at each gate, and why.

One recent paper on applying this pattern at enterprise scale makes a sharper claim still: if the harness's own configuration is treated as a reviewable, version-controlled artefact, governance review collapses into “reading a text file” instead of auditing a bespoke pipeline built by hand for each team [5]. That's a real architectural choice, not a compliance nicety — it determines whether adding a second team or a second agent multiplies your review burden or doesn't.

Figure 1 — harness anatomy. Picture a single closed loop, not a linear pipeline: the agent pursues its *Goal* while operating inside its *Rules*; it reaches a *Gate*; the outcome — approved, rejected, or auto-cleared — is written to the *Log*; the *Log* then feeds back to refine the *Goal* and *Rules* for the next cycle. The loop repeats. Oversight isn't a stage the agent passes through once; it's the shape of the whole system.

Four patterns, with evidence

Approval gates: tiered, not binary

The dominant design in 2026 isn't "approve every action" or "approve nothing." It's tiered: auto-clear the safe and reversible, flag the recoverable-but-consequential, and hard-block only the genuinely irreversible. Anthropic's own usage data on Claude Code makes the case with numbers rather than intuition: only 0.8% of observed agent actions turn out to be irreversible, roughly 80% of tool calls already carry at least one safeguard such as restricted permissions or a mandatory approval step, and auto-approve usage climbs from around 20% among new users to over 40% among experienced ones as trust is earned through use [1]. Anthropic's own conclusion is blunt: requiring approval on every single action "will create friction without necessarily producing safety benefits" [1].

Independent research backs the same shape from a different angle. A dynamic tiered execution framework studied across production workloads routes 55% of tasks through a minimal-overhead path and reserves full, multi-role governance — independent proposal, review, execution and verification — for the 15% of tasks that genuinely warrant it, while also reporting a 67% automated recovery rate on tasks that failed on the first attempt [4]. Mechanically, this is exactly what current vendor tooling implements: a gated call becomes a recorded "interruption" carrying a serialisable run state, which is approved or rejected and then *resumed*, not restarted, from that state — blocking when the risk of proceeding is too high, running in parallel when latency matters more [2]. This is also no longer just a research proposal: enterprise allowlisting for agent tool access, and a general-availability agent control plane, both shipped from a major platform vendor in the first quarter of 2026 [12].

Decision records, not audit trails nobody reads

A gate is only useful if what it produces gets used. The distinction that matters is between an audit trail — filed away for a compliance review that may never happen — and a decision record built to be read by whoever picks up the work next. One governance framework for enterprise AI control planes specifies exactly what such a record needs to contain to do that job: the policy version in force, the attributes that were evaluated, the allow/block/modify decision itself, a reason code, and a timestamp [6]. That's a small, boring schema, deliberately so — it's designed to be resolved by a machine or a human without needing to reconstruct the reasoning from scratch.

This is where the "harness as code" argument from the previous section pays off directly: when the harness's configuration and its decision log are both structured, reviewable artefacts, the record isn't an add-on. It's the same object doing double duty as both governance evidence and the thing the next step actually reads [5][6].

Cross-agent handoff

Handoff — between two agents, or between an agent and a human reviewer — is where context most often gets lost, unless there's a structural carrier for state. Two pieces of shared infrastructure now exist specifically to solve this. The Model Context Protocol's November 2025 specification revision added task-based workflows with status polling and result retrieval across multiple states, and the public registry backing it grew 407% in the months following its initial onboarding batch — evidence this is maintained infrastructure now, not a one-off integration [7]. Separately, Google's Agent2Agent protocol, at version 0.3, hands work between independent agents through an "Agent Card": one agent can expose itself as a discoverable sub-agent to another, regardless of which platform either was built on [8].

Figure 2 — cross-agent handoff. Read it left to right: Agent A completes its portion of the work and reaches a *Gate*; at that gate, it writes a *Decision record* — the same policy-version, decision, reason-code, timestamp fields described above [6]; Agent B, or a human reviewer, reads that record and resumes from it, rather than re-deriving Agent A's reasoning from the raw conversation. What survives the handoff is the record, not the transcript.

Cost and attribution

The fourth pattern is the one sceptical CTOs usually raise last, and it depends on the first three existing at all: you cannot attribute cost or value to autonomy you haven't gated and logged. The FinOps Foundation's guidance on AI token economics lays out three progressively more precise attribution methods — disciplined API-key ownership, provider-native cost-allocation features, and metadata-tagging proxy layers that attach a user, session, feature or cost centre to every call — and warns that token-only accounting under-reports true cost, since the infrastructure wrapped around the tokens typically adds a further 40–60% on top of raw token spend [11]. That attribution data is exactly what's missing at most organisations today: McKinsey's 2026 AI Trust Maturity survey finds that governance and agentic-AI controls lag behind data and technology maturity across every region surveyed [9]. You cannot show where autonomy is paying off, or where it isn't, without the same gates and records that critics dismiss as overhead.

Objections

“Gates slow us down.” Only if every gate is built the same way. The tiered pattern above exists precisely to avoid that: block on the genuinely high-risk, irreversible calls, and run guardrails asynchronously or auto-clear everywhere else [2]. Anthropic’s data shows that once trust is earned, the majority of everyday actions are already auto-approved rather than gated at all [1].

“Nobody reads the log.” That’s true of an audit trail nobody was ever going to open. It isn’t true of a decision record built for the next agent or reviewer to resume from — which is the specific design goal of the schema in the cross-agent handoff pattern [6][5]. The complaint is really an argument against building the wrong kind of log, not against logging.

“This is just more process.” Process without data is bureaucracy; process that produces cost and attribution data a CTO can actually act on is architecture. The gate is what makes that data trustworthy in the first place — without it, “autonomy paid off” is a guess, not a number [11][9].

What to do Monday

None of this requires a platform migration to start.

1. Pick one live workflow and write down its harness explicitly — goal, rules, the specific points where it must gate, and the schema its log will use — before adding a second agent to it.
2. Tier the gates rather than defaulting to “approve everything”: auto-clear reversible actions, flag the consequential-but-recoverable, and hard-block only what can’t be undone [1][4].
3. Standardise the decision-record schema — policy version, decision, reason code, timestamp — before you scale to a second agent, a second team, or a handoff to a human reviewer [6].
4. Wire cost attribution to the same record, using the same events and the same tags, so the data a gate already captures becomes your autonomy ROI review rather than a separate finance exercise [11].

The harness — not the underlying model, and not a compliance sign-off bolted on afterwards — is the architecture decision that determines whether agent deployments compound or drift. In 2026, that’s no longer a theoretical distinction. It’s the difference between the business

functions still capped at roughly 10% scaled adoption and the ones building on top of what already works [9].

Sources

- [1] Anthropic, “Measuring AI agent autonomy in practice,” 18 Feb 2026 — <https://www.anthropic.com/research/measuring-agent-autonomy>
- [2] OpenAI, “Guardrails and human review” / “Human-in-the-loop — OpenAI Agents SDK,” 2025–2026 — <https://developers.openai.com/api/docs/guides/agents/guardrails-approvals> ; https://openai.github.io/openai-agents-python/human_in_the_loop/
- [3] Zhou, Chai, Chen et al., “Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering,” arXiv:2604.08224, 9 Apr 2026 — <https://arxiv.org/abs/2604.08224>
- [4] Kai Pan, Rong Hou, “Beyond Autonomy: A Dynamic Tiered AgentRunner Framework for Governable and Resilient Enterprise AI Execution,” arXiv:2605.10223, May 2026 — <https://arxiv.org/abs/2605.10223>
- [5] George Juraj Salapa, “Applying Anthropic Primitives at Large Enterprises: Harness Paradigm for Knowledge Work,” arXiv:2608.20622, 20 Aug 2026 — <https://arxiv.org/abs/2608.20622>
- [6] Roopam W. Sure, “AGL-1: The Enterprise AI Governance Layer as a Control Plane for Trusted Enterprise Intelligence,” arXiv:2607.03516, 3 Jul 2026 — <https://arxiv.org/abs/2607.03516>
- [7] Model Context Protocol Blog (Anthropic), “One Year of MCP: November 2025 Spec Release,” 25 Nov 2025 — <https://blog.modelcontextprotocol.io/posts/2025-11-25-first-mcp-anniversary/>
- [8] Google Cloud Blog, “Agent2Agent protocol (A2A) is getting an upgrade,” 1 Aug 2025 — <https://cloud.google.com/blog/products/ai-machine-learning/agent2agent-protocol-is-getting-an-upgrade>
- [9] McKinsey & Company, “State of AI trust in 2026: Shifting to the agentic era,” 2026 — <https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-forward/state-of-ai-trust-in-2026-shifting-to-the-agentic-era>
- [10] Stanford HAI, “The 2026 AI Index Report,” 2026 — <https://hai.stanford.edu/ai-index/2026-ai-index-report>
- [11] FinOps Foundation, “Tokenomics: Managing AI Value in SaaS Model Token Costs,” updated 3 Jun 2026 — <https://www.finops.org/wg/token-economics-saas/>
- [12] GitHub Changelog, “Copilot CLI supports custom registry based MCP allowlists” (16 Apr 2026) / “Enterprise AI Controls & agent control plane now generally available” (26 Feb 2026) —

<https://github.blog/changelog/2026-04-16-copilot-cli-supports-custom-registry-based-mcp-allowlists/> ; <https://github.blog/changelog/2026-02-26-enterprise-ai-controls-agent-control-plane-now-generally-available/>

Published by an independent AI-strategy consultant. Word count (body): 1,849. Sources: 12, all 2025–2026.