The Two-Layer Black Box

Operator Visibility, Commercial Secrecy, and a Minimum Disclosure Set for Accountable Autonomous AI Agents

Author: Tatsuya Shimomoto **Affiliation:** Independent researcher **ORCID:** 0009-0002-6168-4162 **Version:** v1 (2026-05-23) **DOI:** 10.5281/zenodo.20355907 (Zenodo — concept DOI, resolves to the latest version) **Companion repository:** Agent Attribution Practice (AAP), DOI 10.5281/zenodo.20251893 (v0.3.0) **Companion paper:** *Distributing Accountability, Not Capability* (Shimomoto 2026b), DOI 10.5281/zenodo.20353789

Abstract

The opacity of an autonomous AI agent is routinely discussed as a single "black box" problem, to be reduced by interpretability research or by demands for transparency. This paper argues that the opacity has two architecturally distinct causes that the discourse conflates, and that the conflation is what makes the transparency debate unwinnable. One cause is technical: behavior internalized into model weights cannot be read, versioned, or reverted without retraining. The other is commercial: the human-built layer around the weights — system prompts, rules, tool definitions, the agent loop — is technically inspectable but kept proprietary because it is a competitive differentiator. The two causes admit different responses, and treating them as one problem routes every transparency argument into an irresolvable safety-versus-secrecy collision.

The paper makes three contributions, drawn from the implementation history of a single agent and re-expressed as harness-neutral judgments. First, it grounds the opacity problem in an *accountability gap*: a text-based prohibition on a capability that physically exists is a signpost, not a control, and enforcement admits a *prohibition-strength hierarchy* — absence, then scaffolding-layer enforcement, then untrusted-content boundary — most published safety work occupying only the weakest layer. Second, it separates the *two-layer black box*: technical internalization (Layer 1, largely permanent) from commercial secrecy of scaffolding (Layer 2, contingent on market structure). Third, it resolves Layer 2 with a *minimum disclosure set*: operator visibility is decoupled from public disclosure, and the minimum that makes post-incident causal tracing possible — which scaffolding version was active, which inputs reached the agent, which approval gated the version — is disclosable without publishing the scaffolding text.

The resolution is implementable with current technology; it waits on neither an interpretability breakthrough nor a restructuring of markets and law. It complements existing AI risk-management and management-system standards by naming the operator-visibility floor those standards presuppose. The open questions — where exactly the disclosure line falls, and how the technology–society timescale gap is to be managed — are the research agenda.

Keywords: AI accountability; black box; scaffolding; minimum disclosure; causal traceability; prohibition hierarchy; AI governance; autonomous agents.

---

1. Introduction

An autonomous AI agent that acts on the world — posts content, calls an external service, edits a record — eventually causes an incident, and someone has to explain it. The agent paradigm that makes this possible — a model that reasons and then acts through tools (Yao et al. 2022), now a standard practitioner pattern (Anthropic 2024; OpenAI 2025) — is also what creates the explanatory burden. The explanation requires reconstructing why the agent did what it did. For most deployed agents, that reconstruction fails: the final

output and the list of external calls are logged, but the chain leading to the decision is not recoverable. The system is a black box.

The dominant framing treats this opacity as a single problem with a single class of solution. Interpretability research aims to read the model's internal computation. Transparency advocates demand that builders disclose what their agents do. Both are locally reasonable, and both run aground on the same point: the opacity of an agent does not have one cause. It has two, and they are different in kind. Conflating them is what keeps the transparency debate stuck — every argument for disclosure collides with a reason for secrecy that belongs to a different layer of the problem.

This paper separates the two layers. The first is *technical*: capabilities, reasoning patterns, and learned values dissolved into model weights during training. This layer is opaque by construction; it cannot be read by inspection, versioned, diffed, or reverted without retraining. The second is *commercial*: the human-built layer around the weights — system prompts, personas, rules, tool definitions, retrieval indices, the agent loop, safety gates, the runtime harness — which AI-safety discourse calls *scaffolding* (beren 2023). Scaffolding is technically inspectable. Stored as files and tracked in version control, every change to it is visible. Yet in commercial deployments it is usually hidden, because prompt design and behavioral tuning are product differentiators with no incentive to disclose. The first layer is invisible because we cannot see it; the second is invisible because someone chose not to show it.

The distinction matters because the two layers have different resolutions. The technical layer is, for now, a research problem with no near-term operational answer. The commercial layer is a structural problem — a collision between a safety requirement (behavior should be traceable) and a market requirement (differentiators should be protected) — and structural problems admit structural resolutions. The paper's central claim is that the commercial layer can be resolved, today, by separating two questions the discourse runs together: whether the *operator* who runs an agent can read its scaffolding, and whether the *public* can. The first must always be answerable; the second is a legitimate commercial choice. Between them lies a *minimum disclosure set* — the smallest body of evidence that makes post-incident causal tracing possible without publishing the scaffolding's full text.

This claim does not stand alone. It rests on a prior diagnosis about why agents need traceable behavior at all, and on a requirement about what traceability concretely demands. The paper develops all three in sequence, drawing on a sister body of work: the implementation and operational history of a single LLM-based agent, recorded as architecture decision records in a public repository (Shimomoto 2026a), and the trilogy of essays (Articles 1–3 of that repository's seven-essay narrative spine; Shimomoto 2026a, `inspiration.md`) from which those records were extracted. The argument is offered as a *position*, not as an empirical study; the repository is the empirical substrate. A companion position paper (Shimomoto 2026b) develops the architectural follow-up — the four Business AI Quadrants and the design/operation Phase Separation — that the surface-layer diagnosis here precedes.

The remainder is organized as follows. Section 2 establishes the accountability gap: why a text prohibition on a reachable capability is not a control. Section 3 develops the prohibition-strength hierarchy that the gap implies. Section 4 states what causal traceability concretely requires of the artifact layer. Section 5 separates the two-layer black box. Section 6 resolves the commercial layer with the operator/public decoupling and the minimum disclosure set. Section 7 situates the framework against existing standards. Section 8 closes with open questions. The thesis carried throughout is that what organizations have refined over centuries is not the distribution of *capability* but the distribution of *accountability*, and that the black-box problem is, at its tractable layer, an accountability problem wearing the costume of a transparency problem.

2. The Accountability Gap: Why Signposts Do Not Enforce

A prohibition written on a barrier that can be crossed carries no force. The power of a norm lies not in being recorded but in being enforceable. Current AI-agent governance is largely in the recording phase: "do not produce harmful content" written into a system prompt, ethics guidelines published as documents, safety committees convened. The instructions multiply while the barrier they describe remains crossable.

The problem is old. Plato's *Republic* poses it through the Ring of Gyges: a shepherd finds a ring that makes him invisible — capability with no observation and no record — and uses it to seize a kingdom. The thought experiment asks whether anyone would keep the rules knowing no one was watching and nothing was logged (Plato, *Republic*, Book II). That is the situation of an unconstrained agent: capable, unobserved, accountable to nothing. Hobbes (1651) framed the same problem in *Leviathan* as a question about contracts — if defection is undetectable, defection is rational — and answered it with what amounts to reputation-based access control: defectors are excluded from future cooperation. Both diagnoses locate the operative variable not in the rule but in the consequence of breaking it.

Engineering practice already encodes this. The enforcement patterns that actually constrain behavior fall into three kinds: *physical constraint* (make the action impossible — sandboxing, permission models, access control), *consequence* (make it costly — penalty-based conditioning), and *internalized value* (make the actor not want to — value alignment). Each is a real mechanism. Set against them is a fourth arrangement that is not a mechanism at all: the *signpost* — a text-based rule with no enforcement attached. In a codebase it is the comment that says *do not do this*. It does not constrain; it documents.

Signposts persist for a reason unrelated to prevention. In organizations, governance-by-documentation produces a policy that costs almost nothing to write and enforces almost nothing, but creates a record that measures were taken. Its real function is *indemnification*, not prevention: when something goes wrong, the documented rule shifts liability onto whoever crossed the line. AI guardrails written as prompt clauses follow the same pattern — thin practical effect, durable paper trail, blame redirected to the user who "misused" the system rather than to the builder who left the barrier crossable.

The operational consequence is sharp. A rule written for a language model is followed *probabilistically*: under normal conditions an instruction in a system prompt or convention file is honored perhaps half to four-fifths of the time (an operational observation; Shimomoto 2026a, ADR-0002), and under adversarial pressure — prompt injection, or merely a debugging session in which an operator asks the agent to do the forbidden thing — compliance drops further. A constraint enforced outside the model, in the execution layer, fires *deterministically*: it intercepts the action before it occurs, on every matching input, and its decision is derivable from configuration rather than from the model's disposition. The gap between the two is the gap between a signpost and a wall.

This is the accountability gap: the distance between governance that records intentions and governance that enforces them. Closing it does not require new theory. Every engineering organization already runs approval workflows, deployment gates, audit logs, and review processes that share three properties — approval is recorded, responsibility is assignable, changes are auditable. What organizations have refined over centuries is not the distribution of capability but the distribution of accountability (Shimomoto 2026a, `thesis.md`). Agents are built today with almost none of it. The next section asks what closing the gap looks like when the closure is taken seriously as an ordering over enforcement mechanisms, rather than as a single switch.

---

3. The Prohibition-Strength Hierarchy

If a signpost is the weakest possible prohibition and a physical wall the strongest, then enforcement is not binary but graded. The implementation history of the agent that grounds this paper surfaced a three-level ordering, derived not from theory but from the failure of weaker mechanisms under operational pressure (Shimomoto 2026a, ADR-0002).

The strongest level is *absence*. A dangerous capability that is never implemented cannot be invoked, misconfigured, or hijacked. An agent whose core ships no shell execution, no arbitrary outbound network access, and no filesystem traversal outside a narrow data directory presents no attack surface for those capabilities, because the surface does not exist. Prompt injection cannot grant an ability the harness was never built to have (Shimomoto 2026a, ADR-0001). Absence has a property no other level shares: its audit test is a search. The question "does capability X exist?" is answerable by grepping the codebase, not by reasoning about whether a guard will hold. Its only failure mode is a later change that adds the capability — a failure visible in code review.

The middle level is *scaffolding-layer enforcement*. Some capabilities cannot be made absent: an agent must read certain files, call a service, observe tool output. For these, the prohibition is enforced in the execution layer — a pre-execution hook that blocks a matching action, a structural quarantine that prevents external content from entering the model's context at all, an adapter gate on the single outbound channel. All three share the defining property that the blocking decision *does not consult the model*; correctness is verifiable by reading configuration and the gating code (Shimomoto 2026a, ADR-0002). This level is weaker than absence — a matcher can have gaps, a gate can be misconfigured, a novel path can bypass it — but its residual risks are *enumerable*, and a violation is *diagnosable*: did the gate fire; if not, what did the matcher miss; if so, how was it bypassed.

The weakest level is the *untrusted-content boundary*. When content must reach the model, it is wrapped and annotated as untrusted, and accumulated state — logs, knowledge stores, the agent's own distilled output — is validated against a forbidden-pattern list on read-back, failing closed, and treated as untrusted regardless, because a summary of untrusted input inherits the taint of its sources (Shimomoto 2026a, ADR-0003). Here the constraint is honored probabilistically: the model is asked to treat marked content with suspicion, and may or may not comply. The limit of this level is visible in memory-injection-class attacks, in which malicious content placed in an agent's stored state steers its later behavior. In operation, the defense that actually held against such an attack was not a probabilistic trust-weighting of sources — which looked like a defense on the schema but did no discriminating work in production — but a structural quarantine one level up, keeping external content out of the model's context entirely (Shimomoto 2026a, ADR-0002). The lesson is the hierarchy's own: where the weakest level cannot hold, the prohibition has to climb. This is the level at which most published AI-safety interventions operate — refusals trained into weights, constitutional clauses, prompt-level instructions — and it is the level whose compliance cannot be guaranteed.

The hierarchy is *absence* > *scaffolding enforcement* > *untrusted boundary*, and its load-bearing rule is procedural: walk it from the top, and drop to the next level only when the current one genuinely cannot hold the capability (Shimomoto 2026a, glossary, "prohibition-strength hierarchy"). The diagnostic value of the ordering is that it makes a common error visible. Most AI-safety work lives at the weakest layer — probabilistic text constraints — without acknowledging that the stronger layers exist and are usually available (Shimomoto 2026a, `thesis.md`). A claim of the form "the agent is instructed not to do X" is, in the hierarchy's terms, a claim about the weakest level, frequently made where the strongest level was achievable. Two clarifications bound the hierarchy. It is not a claim that the weakest level is worthless: for *continuous judgments* — tone, care, contextual appropriateness — the model layer is the only available layer, and the hierarchy concerns *categorical prohibition*, not values. And the choice of a lower level is always

a concession, never a default: for every gated capability, the question "could this have been made absent instead?" remains live.

The hierarchy is the structural answer to Section 2′s accountability gap. It does not close the gap by exhortation; it orders the mechanisms that close it by how reliably they fire. With enforcement ordered, the next requirement is the one that makes enforcement *answerable after the fact* — the ability to reconstruct what an enforced or unenforced decision actually was.

---

4. Causal Traceability as a Build-Time Requirement

Enforcement decides what an agent may do. Traceability decides whether anyone can explain what it did. The two are distinct: an agent can be tightly constrained and still opaque about which constraint was active, on which input, approved by whom. After an incident, the explanation requires reconstructing *which version of the agent did what, in response to what input, producing what output, approved by whom* — and that reconstruction is not achievable at runtime. It has to be designed into the artifact layer before the incident (Shimomoto 2026a, ADR-0006).

The economics force the investment upstream. A widely shared lesson from site-reliability practice is that the cost of recovering from an incident dwarfs the cost of building so the incident does not occur — often by an order of magnitude, once investigation, recovery, communication, audit response, and the slow rebuilding of trust are counted. Preventive structure, by contrast, is paid incrementally inside ordinary development. Even discounted by incident probability, the upstream investment usually comes out smaller in expectation. The allocation that minimizes total cost places structure upstream, and the determinant of that conclusion is not ideology but the asymmetry of incident costs — a constraint closer to physics than to philosophy (Shimomoto 2026a, ADR-0006).

What "structure upstream" requires, concretely, is a small set of artifacts that already exist at build time. The episode log records every input the agent saw, append-only, one line per event — append-only as a *system property*, because rotating or truncating it destroys the only complete record of what the agent actually received. Behavior-modifying components — identity, rules, the agent's normative definition, skills — are *version-pinned*, so that "which rules were active when the incident happened" has a definite answer rather than the moving target of a file overwritten in place. An audit trail records every approval decision with its reason and its approver. Architecture decision records preserve why the current structure exists. An adapter call log records the request/response pairs on the single external channel. These are not runtime observability; metrics and traces capture *that* something happened, while causal tracing needs the *why*, which lives in the versioned artifact state at the time (Shimomoto 2026a, ADR-0006).

The artifacts are not novel. They are structurally identical to what organizations have used for human decisions for centuries: segregation of duties, the four-eyes principle, least privilege, internal controls, the change advisory board, the audit log. Each produces a chain that can be walked backwards from an incident to a decision. The convergence of organizational governance, software engineering practice, and agent design on the same set of structures is not decorative; the same incident-cost asymmetry that drove humans to adopt these practices applies identically to agents (Shimomoto 2026a, ADR-0006). One artifact also fixes the *endpoint* of the chain: each agent process is bound to exactly one identifiable human who is accountable for it — the approver of behavior-modifying changes, the owner of incident response, the party who answers for the agent's behavior. "The team owns the agent" diffuses accountability into organizational abstraction; "the agent is autonomous" ends the chain at "the model did it"; "AI supervises AI" circles among probabilistic systems without reaching a human. Multiplexed accountability is diffused accountability, and the chain must terminate at a named person (Shimomoto 2026a, ADR-0008, *experimental*).

Traceability done this way changes what a postmortem can be. Instead of the LLM's post-hoc rationalization — a narrative, not a cause — the operator can state which external surface was touched, which decision logs led to the output, which scaffolding version was active, and who approved it. Causal attribution distributes across the structure, and responsibility distributes with it. Without that structure, blame defaults to the on-call engineer by elimination — the person responsible for a black box is classified, after the fact, as the one who failed to watch, because there was no defined place to watch from. This is the structural arrangement Elish (2019) named the *moral crumple zone*: responsibility for an autonomous system concentrates on a human whose actual control over the runtime decisions was limited. Seeking causes in the system instead is the site-reliability discipline of the blameless postmortem (Shimomoto 2026a, ADR-0006). This is the requirement that the next two sections must satisfy. Traceability demands that the active scaffolding version be *readable*. Whether it is readable is exactly what the black-box problem puts in question.

---

5. The Two-Layer Black Box

Causal tracing requires reading the scaffolding that was active. The black-box problem is that, in commercial deployments, the scaffolding cannot be read — and the reason it cannot be read is not the reason usually given. The opacity has two distinct causes, and the discourse treats them as one (Shimomoto 2026a, ADR-0007).

The first cause is genuinely technical. Through pre-training and reinforcement learning, a model's language ability, commonsense reasoning, and learned values are dissolved into its weights. This layer — Layer 1, the model internals — is opaque by construction. It cannot be read by inspection, versioned, diffed, or reverted without retraining; a postmortem cannot point at the line that caused a behavior, because there is no line. Chain-of-Thought is the clearest illustration of the layer's movement. It began as an external prompt — *let's think step by step*, a string outside the model — and was internalized into the weights of reasoning models to capture performance gains that external prompting could not deliver. What was once inspectable scaffolding became opaque internals. The internalization was a performance-first design choice with a visibility cost, not a technical inevitability, but its result is the same: behavior that has crossed into Layer 1 is, for now, beyond operational inspection.

The second cause is not technical at all. Outside the weights lies *scaffolding*: system prompts, persona definitions, rules, tool definitions, retrieval pipelines, the agent loop, safety gates, the runtime harness. This layer — Layer 2 — is *technically visible*. Store it in files, track it in version control, and every change is inspectable. When scaffolding is properly managed, the model is an interchangeable inference engine and the agent's essence lives in the scaffolding: swap the model, keep the scaffolding, and the agent behaves the same way. Yet Layer 2 is, in practice, hidden — not because it cannot be read, but because reading it is commercially disadvantageous. Prompt design and behavioral tuning are product differentiators; published scaffolding is scaffolding a competitor can copy. The invisibility of Layer 2 is a market fact wearing the appearance of a technical one.

The collision is structural. Safety evaluation needs scaffolding to be visible — research has found that scaffolding and other post-training enhancements can yield large compute-equivalent gains, upward of 5x for most enhancements studied and beyond 20x for some, so that evaluating a model in isolation, without its scaffolding, understates its effective capability (Davidson et al. 2023). The commercial environment needs scaffolding to be hidden. *It should be visible for safety; it must stay hidden for business* (Shimomoto 2026a, ADR-0007). The current black box sits at the equilibrium of those two forces. That even an AI company among the most committed to safety had not published its agent's scaffolding — made plain when a 2026 accidental source-code disclosure laid the code bare — is not a story about that company's choices;

it is evidence that the competitive environment forecloses publication regardless of safety commitment (Shimomoto 2026a, ADR-0007).

One reason the conflated debate persists is that its participants are not a representative sample. Developers at the frontier can often infer causes from outputs and adjust prompts themselves, so to them approval gates and audit trails look like inefficient ritual their own skill can substitute for; the operators, auditors, and incident responders who depend on those structures are rarely on the conference stage. "Tear down the structures," rational from one cross-section, reads as "do not tear down the structures we depend on" from another — the disagreement is about field of view, and separating the two layers is what widens it.

This is not a charge that commercial secrecy is illegitimate. Protecting a differentiator in a competitive market is rational, and denying that rationality resolves nothing. The problem is in the structure — the point where market rationality and a safety requirement collide — not in the actors. And that is precisely why conflating the two layers makes the transparency debate unwinnable. An argument for disclosure aimed at Layer 1 is a demand for an interpretability breakthrough that does not yet exist. The same argument aimed at Layer 2 is a demand that firms surrender competitive advantage. Run together, the two produce a debate in which "black boxes are dangerous, make them transparent" meets "transparency is technically impossible and commercially suicidal," and neither side is wrong about its own layer. Separating the layers is the precondition for any tractable move: it locates *where intervention is possible*. Layer 1 is, for now, where it is not. Layer 2 is where it is — and the next section is what intervention there looks like.

---

6. Operator Visibility vs Public Disclosure: The Minimum Disclosure Set

The resolution of Layer 2 turns on a distinction the transparency debate omits. "Can the scaffolding be read?" is two questions, not one: *can the operator who runs the agent read it*, and *can the public read it*. Conflating them is what makes commercial secrecy look like a complete defense of opacity, when it defends only one of the two (Shimomoto 2026a, ADR-0007).

The two questions have different answers. Operator visibility is non-negotiable. Anyone running an agent must be able to read the full scaffolding of the instance they run — there is no scaffolding the operator is trusted to deploy but not trusted to read. The reason is structural, not ethical: if the operator cannot read the scaffolding, no human in the loop can review changes to it, and the approval gate and causal traceability of the preceding sections become unenforceable. Optional operator visibility is not visibility. Public visibility is the separate question, and on it the position is permissive: commercial operators may have legitimate reasons to redact scaffolding from public view, and requiring full public disclosure is overreach. What the framework forbids is narrow and specific — using commercial secrecy, the legitimate reason to withhold scaffolding from the *public*, as cover for also withholding it from the *operator*, which is what dissolves the accountability chain (Shimomoto 2026a, ADR-0007).

Concretely, scaffolding must be *materialized as files* (no behavior-shaping component living only in a runtime variable or a proprietary API response), *version-controlled* (changes diffable, revertible, attributable), *operator-inspectable*, and *gated at write time* through the approval step. This draws a boundary the black-box discourse usually leaves implicit: between *healthy scaffold dissolution*, in which scaffolding melts into clear conversational patterns and readable files, and *unhealthy internalization*, in which scaffolding is absorbed into weights where it cannot be read back. The first preserves accountability; the second is Layer 1 capture of what was Layer 2, and it is what the operator-visibility requirement forbids (Shimomoto 2026a, ADR-0007).

With the two questions separated, the constructive resolution follows. The answer to "visible versus hidden" is neither *publish everything* nor *hide everything*; it is defining the *minimum set* that makes causal tracing

possible (Shimomoto 2026a, ADR-0007). That set is exactly what Section 4 required: which scaffolding version was active at the time of the incident, which inputs reached the agent, and which approval record gated the active version — the three of Section 4′s artifacts a third party needs to reconstruct a specific incident, as distinct from the full artifact layer the operator maintains. An organization can disclose this minimum set — for an audit, an incident review, a regulator's inquiry — *without* disclosing the scaffolding's full text. The competitive differentiator stays protected; the accountability chain stays intact. Below this line there is no accountability chain at all; above it, the precise placement of the disclosure boundary is operator-specific and is itself a judgment to be recorded.

This resolution has a property the technical-layer debate cannot offer: it is implementable now. It does not wait for an interpretability advance that would let us read Layer 1, and it does not wait for markets or legal frameworks to restructure around AI. An individual operator with no commercial reason to hide scaffolding satisfies the requirement by default — which is the existence proof that the minimum disclosure set is achievable, since the obstacle was never technical. A commercial operator satisfies it by separating what must be disclosed for tracing from what may be withheld for competition. The timescale argument makes the point urgent rather than optional. New technologies reach broad adoption only after social structure reorganizes around them, and that reorganization runs decades to centuries behind the technology — movable-type print culture took centuries to remake the institutions of knowledge (Eisenstein 1979), and electricity and the internet showed decade-scale lags between deployment and institutional change. AI's technical change is fast; the change in law, regulation, and organizational practice is not. There will be a gap period, and agents have to function within it. The minimum disclosure set is a mechanism for functioning within existing structures during that gap, rather than a demand that the structures change first.

---

7. Relationship to Existing Frameworks

The framework here is not a substitute for AI risk-management and management-system standards; it is the judgment layer those standards presuppose. The NIST AI Risk Management Framework (NIST 2023) organizes risk activities into govern, map, measure, and manage functions; ISO/IEC 42001 (ISO/IEC 2023) specifies an AI management system. Both ask an organization to manage risk and document its decisions. Neither, by design, decides for a given agent whether a prohibition was enforced at the strongest available level, what the minimum disclosure set contains, or who the accountable operator is. Those are the per-deployment judgments that populate the standards' templates.

The contributions map onto that gap directly. The prohibition-strength hierarchy (Section 3) gives a manage-function activity a strength ordering, so that "we instructed the model not to" is recognizable as the weakest available control rather than as a control of unspecified strength. Causal traceability (Section 4) names the artifact substrate that makes a measure or audit activity reconstructable rather than narrative. The minimum disclosure set (Section 6) gives a concrete, partial answer to a transparency requirement — disclose this much for tracing, withhold the rest for competition — that a standard states as a goal but does not operationalize. The two-layer separation (Section 5) is a map-function input: it directs a risk-characterization activity to distinguish technical opacity, which it cannot remediate, from commercial opacity, which it can. The framework records the judgment; the standard records that a judgment was made.

The companion paper (Shimomoto 2026b) develops the architectural layer downstream of this one: a four-quadrant decomposition of business AI work and a design/operation Phase Separation, which determine *whether* an autonomous loop is the right architecture before the accountability mechanisms here are even brought to bear. The surface-layer diagnosis of this paper — accountability gap, prohibition hierarchy, two-layer black box, minimum disclosure set — is the foundation that the architectural triage

presupposes. Together the two papers distill the same seven-essay discovery trajectory (Shimomoto 2026a, `inspiration.md`): the trilogy here, the architectural follow-up there.

---

8. Conclusion and Open Questions

The black-box problem of autonomous AI agents is two problems wearing one name. One is technical opacity — behavior internalized into weights, beyond operational inspection for the foreseeable future. The other is commercial secrecy — scaffolding that is technically readable but withheld because it is a competitive differentiator. Conflating them makes the transparency debate unwinnable, because every argument for disclosure that is true of one layer is false of the other. Separating them locates where intervention is possible, and at the layer where it is possible — the commercial one — the intervention is structural and available now.

Three observations close the paper.

The tractable layer is an accountability problem, not a transparency problem. The demand "make the agent transparent" is unanswerable for Layer 1 and overreaching for Layer 2. The answerable demand is narrower: that the operator can read the scaffolding they run, and that a minimum disclosure set — active scaffolding version, inputs received, gating approval — can be produced after an incident. Transparency to the public is optional; traceability for accountability is not. The reframing is the contribution: what looked like a problem about seeing inside the box is a problem about who can be redirected to when the box fails.

The resolution does not wait. The minimum disclosure set is implementable with current technology, and the existence proof comes from operators who have no commercial reason to hide scaffolding and therefore satisfy the requirement by default. It waits on neither an interpretability breakthrough for Layer 1 nor a restructuring of markets and law. It is a way to operate accountably within existing structures during the decades-long gap before those structures catch up to the technology.

Implementation dissolves; judgment persists. The specific mechanisms that carry these commitments — append-only JSONL logs, version-pinned scaffolding files, pre-execution hooks, approval queues — are not durable; they will be replaced. What persists is the judgment they encode: that prohibition has a strength ordering and the weakest level is usually chosen where a stronger one was available; that opacity has two causes and only one is technical; that operator visibility is the floor below which no accountability chain exists. These are not implementations. They are decisions about who is answerable when an agent's behavior cannot otherwise be explained.

Open question 1. Where the disclosure line falls. The minimum disclosure set names three elements that must be disclosable for tracing; the exact boundary between "disclose for accountability" and "withhold for competition" is operator-specific and currently a design judgment. What structural criterion makes the boundary inspectable rather than asserted, and transferable across organizations, is open.

Open question 2. The technology–society timescale gap. The framework is a mechanism for functioning within existing structures during the gap period, but it takes no position on which structures should ultimately change. Which accountability requirements are artifacts of the gap and which are durable — and how to tell the difference before the structures have finished reorganizing — is unresolved.

Open question 3. The accountability endpoint at scale. Binding each agent to one named accountable operator (Section 4) is stated as experimental (Shimomoto 2026a, ADR-0008). Whether one-human binding scales to enterprise deployment, continuous operation across time zones, and decisions that cross multiple stakeholders — and what the handoff and revocation protocols are — needs operational testing the single-agent history behind this paper cannot supply.

The paper separates what was conflated. It does not close the questions that separation opens; it makes them stable enough to ask.

References

Anthropic (2024). *Building Effective Agents*. https://www.anthropic.com/research/building-effective-agents

beren [pseud.] (2023). *Scaffolded LLMs as Natural Language Computers*. LessWrong. https://www.lesswrong.com/posts/43C3igfmMrE9Qoyfe/scaffolded-llms-as-natural-language-computers

Davidson, T., Denain, J.-S., Villalobos, P., & Bas, G. (2023). *AI Capabilities Can Be Significantly Improved Without Expensive Retraining*. arXiv:2312.07413. Eisenstein, E. L. (1979). *The Printing Press as an Agent of Change*. Cambridge University Press.

Elish, M. C. (2019). Moral Crumple Zones: Cautionary Tales in Human-Robot Interaction. *Engaging Science, Technology, and Society* 5: 40–60. https://doi.org/10.17351/ests2019.260

Hobbes, T. (1651). *Leviathan.*

ISO/IEC (2023). *ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system.* International Organization for Standardization.

NIST (2023). *AI Risk Management Framework (AI RMF 1.0).* NIST.AI.100-1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

OpenAI (2025). *A Practical Guide to Building Agents*. https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf

Plato. *Republic*, Book II (the Ring of Gyges). Multiple translations.

Shimomoto, T. (2026a). *Agent Attribution Practice (AAP), v0.3.0*. Public repository and Architecture Decision Records. Zenodo. https://doi.org/10.5281/zenodo.20251893. Referenced ADRs: ADR-0001 (Security by Absence), ADR-0002 (Deterministic Prohibition at the Scaffolding Layer), ADR-0003 (Untrusted Content Boundary), ADR-0006 (Causal Traceability), ADR-0007 (Scaffolding Visibility), ADR-0008 (One Agent, One Human). Referenced supporting documents: `thesis.md`, `glossary.md`, `inspiration.md` (seven-essay narrative spine, Articles 1–7).

Shimomoto, T. (2026b). *Distributing Accountability, Not Capability: Phase Separation and the LLM Workflow Quadrant in Autonomous AI Agent Architectures.* Zenodo. https://doi.org/10.5281/zenodo.20353789

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.