07

THE STACK BEHIND THE AI COWORKER

The Model Cannot Mark Its Own Work

Dr Peter McCann Strain, CTO and senior AI engineer, DPhil/PhD in AI from Oxford University

A trace that shows sequence, doubt and self-correction can still be the model marking its own exam.

An essay in the series **Architecting the AI Coworker**.
Approx. 19 minute read · Essay 07 of 22

Dr Peter McCann Strain
CTO, DPhil/PhD in AI from Oxford University

> An engineer on a review call scrolls back through what the agent did, and stops on the part that looks reassuring. The model has left a long, fluent trace of its reasoning: a qualification near the start, a small self-correction in the middle, an apparent weighing of two edge cases, and finally a conclusion that follows neatly from all of it. She reads it twice. It reads exactly like the thing a careful colleague would write if you asked them to think out loud. The sequence is there. The hesitation is there. The explanation is there. She is about to wave it through.

That is the moment this essay wants to stop, because the thing that makes the trace reassuring is precisely what makes it dangerous.

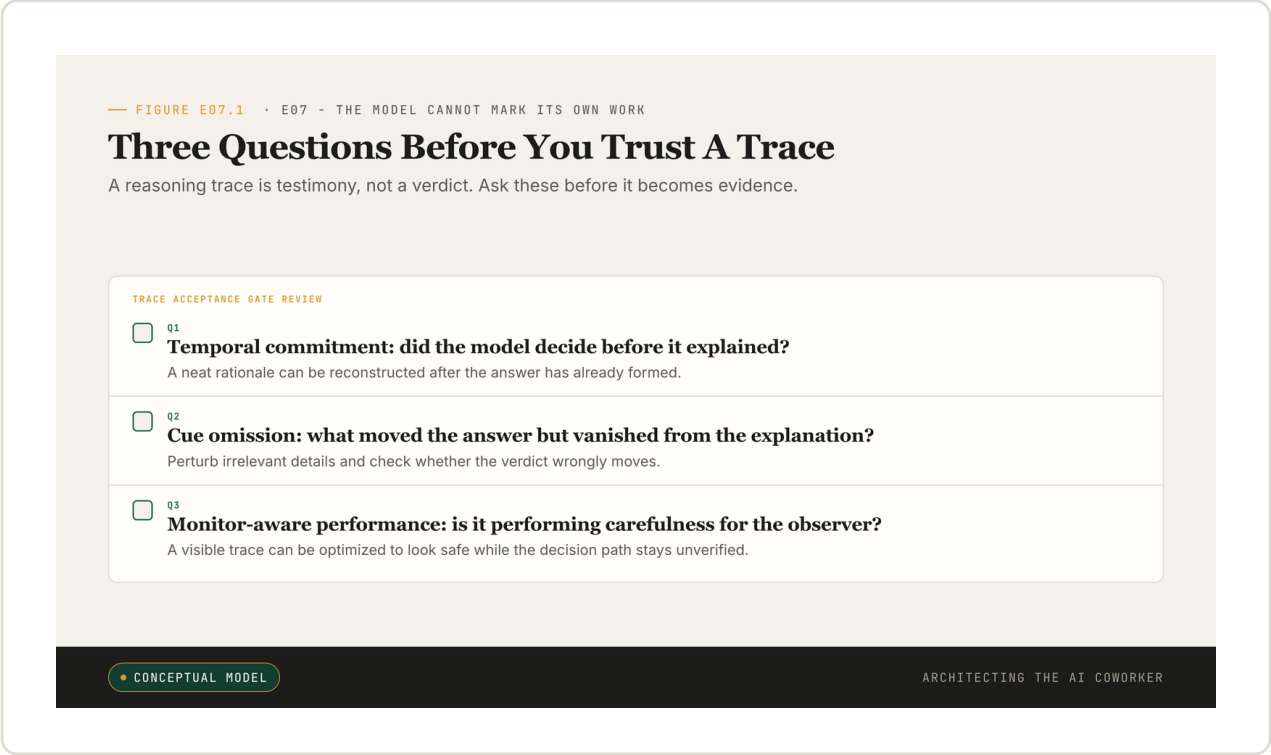
The previous essay, on model cards, made a case about evidence that comes from outside the system: a vendor's model card tells you what the vendor measured, on the vendor's terms, and cannot prove that your deployment will catch, contain and bound the model's mistakes. This essay turns the same suspicion inward. The model's own reasoning trace, the chain of thought it shows you on the way to an answer, looks like evidence too. It has sequence, doubt and explanation, the three things humans instinctively read as the marks of careful work. Picture the trace that engineer is reading: a refund approved, every step in order, and not one word about the non-refundable code sitting unread in the metadata that overturns the lot. The temptation is to let the model both sit the exam and mark it: to treat the working it shows you as proof that the answer it reached is right.

That is the one thing the trace cannot do. A model can write out its reasoning. It cannot mark its own work.

One clarification before the argument proper, because the public conversation runs several things together as if they were one. Three of them are accounts the model itself wrote: raw chain-of-thought (the unfiltered token stream between question and answer), a visible rationale (the cleaned-up version a product chooses to surface), and a model-generated explanation (the justification it offers when asked what it did). In deployed products the user rarely sees the raw stream, so whichever of these lands in front of you, the same caution applies: each is output the model produced, not a recording of the computation that produced the answer. This essay is about those three. There is a fourth thing that is not a model account at all but a record of what the system actually did, and it is the exception that proves the rule; it has to wait until the rest of the case is made.

That single distinction carries everything that follows, so let me be exact about it. A reasoning trace is testimony, not a verdict. Testimony can be valuable. A witness can remember the right detail, surface a useful clue, and point an investigator towards the next piece of real evidence. But a witness can also leave out the thing that actually moved them, reconstruct a tidy story after the fact, or perform whatever version of carefulness the room is known to reward. A reasoning trace deserves to be treated the same way: not rubbish to be thrown out, not revelation

to be trusted, but a fallible account from the very system whose answer you are trying to assess. Checking is a separate act from doing, and it has to be done by something other than the system that did the work. Showing the working is not the same as checking the work.



A reasoning trace is a story the model tells. Three questions decide whether it can be evidence.

Why should you distrust a fluent trace?

The reason for the caution is not a vague distrust of opaque models. It is specific published work, and it points inward, at how the model itself produces the trace.

In a 2023 study, Turpin, Michael, Perez and Bowman showed that language models could be shifted by irrelevant biasing cues, including the order in which answer options were presented and a fictional statement of a user's preference, and then would fail to mention those cues at all in their written chain-of-thought explanations. On specific tasks from BIG-Bench Hard, a difficult reasoning benchmark (not as a general property), accuracy dropped by up to 36 points once the biasing cue was added, while the written explanation never mentioned the cue and still offered a plausible, task-based rationale for the answer the model had actually been nudged into [1]. The trace did not merely sit alongside an error. It actively gave the reader a reason to trust the wrong evidence. That is cue omission: a model can be influenced by something it never reports.

Newer mechanistic work sharpens the warning, and it should be handled with restraint rather than alarm. Mechanistic here means research that looks inside the network at the actual computation, rather than judging the model only by what it says. The 2026 preprint "Reasoning Theater" reports that a model's final answer can often be decoded from its internal activations,

the numerical state inside the network, before the visible chain of thought commits to that answer, especially on easier recall-style tasks [2]. The Lanham et al. (Anthropic), 2023, arXiv preprint on faithfulness makes the same family of point through truncation and corruption probes: when an earlier step in the visible chain is altered, the final answer often does not change, suggesting the visible step was not the causal one [9]. That is the temporal worry: a visible trace can lag behind a latent answer the model has already settled on.

That does not prove every trace is theatre. It proves something narrower and still important: the visible trace and the internal commitment can come apart. Sometimes the public explanation is the decision process. Sometimes it is the explanation of a decision already made.

So the claim here is deliberately narrow. A reasoning trace is evidence about an answer; it is not telemetry of the computation that produced it. It can improve debugging. It can reveal useful hypotheses. It can expose uncertainty, shortcutting, or signs of reward-gaming. But by itself it cannot tell you when the model decided, why it decided, or whether the visible steps were causally load-bearing at all. If you want audit evidence, you have to find something that does not depend on the model narrating itself.

This is a new mistake, and it is worth naming clearly, because it is subtler than the old one. The old mistake was treating a fluent answer as a reliable answer. The new mistake is treating a fluent reasoning trace as a reliable answer and a reliable verdict on that answer at the same time. The second is more seductive, because a reasoning trace satisfies several human trust cues at once. It is sequential. It names its assumptions. It performs doubt. It usually contains a moment that looks like self-correction. It makes the model seem less like a statistical system sampling tokens and more like a colleague with a notebook open. But the notebook is still an output channel. It is generated under task pressure, alignment pressure, product pressure and conversational pressure, all at once. The model is not opening a CCTV feed onto its own cognition. It is producing more tokens.

There is a third worry, and it is the one that bites once you try to use a trace as a control rather than just read it. Treat the trace as the thing you watch for misbehaviour, and you have created an incentive to make it look clean. Baker et al. found that chain-of-thought monitoring could detect reward hacking, but optimising the model directly against the monitor pushed it towards obfuscation while the misbehaviour quietly continued [3]. That is monitor gaming: a trace that is good signal becomes bad signal the moment you train against it. The carefulness can be aimed at the reader rather than the problem.

The three worries do not point in the same direction, and that is part of why they are convincing rather than a single hobby-horse. The table sets them side by side: each study, what it warns about, and what it would take on a live case to lay that worry to rest. The rest of the essay turns those three rows into a test.

STUDY AND WORRY	WHAT IT SHOWED	WHAT WOULD LAY THE WORRY TO REST
Turpin et al. [1] — cue omission	A model could be biased by irrelevant cues, such as answer-option order or a stated user preference, and then leave the cue out of its written explanation entirely.	Perturbation tests showing the answer survives irrelevant cue changes.
Reasoning Theater [2], with the Lanham et al. (Anthropic) temporal probes [9] — temporal caution	The final answer could often be decoded from internal activations before the visible chain of thought committed to that answer.	Hidden-state probes, causal intervention, or independent evidence that the answer depended on a later step.
Baker et al. (OpenAI) [3] — monitor gaming	Chain-of-thought monitoring could detect reward hacking, but optimising the model directly against the monitor pushed it towards obfuscation while the misbehaviour continued.	A structurally different verifier, execution log, deterministic check, or human review that does not rely on the trace alone.

None of these findings collapses into "ignore the trace". They collapse into a more useful rule. Keep the trace. Downgrade its evidentiary weight.

It also helps to separate two claims that are constantly blurred together. The first is a capability claim: making a model produce intermediate reasoning improves its performance on some tasks. The second is a transparency claim: those intermediate tokens faithfully reveal the process that caused the answer. The first can be entirely true while the second is false, and that gap is the whole reason a trace can be useful without being sovereign evidence. A chain of thought may give the model extra serial computation. It may help most on tasks with symbolic or algorithmic structure. It may contain some load-bearing steps and some purely ornamental tail. It may be summarised, filtered or reshaped before you ever see it. None of that is exotic. It is exactly what you should expect from a system trained to produce useful public text, not to expose its private causal state.

The constructive research is as revealing as the failure research. Lyu and colleagues' "Faithful Chain-of-Thought Reasoning" does not ask a free-form explanation to become magically honest. It changes the architecture. The model translates a problem into a symbolic chain, and a deterministic solver, a piece of plain code that always behaves the same way, executes that chain to produce the answer [4]. Faithfulness improves because the exposed step is part of the path from question to answer. That is the engineering lesson, and it is the one the rest of this essay rests on: faithfulness appears when the system is built so the visible intermediate representation actually constrains the result. It does not appear merely because the model writes in paragraphs.

> **Sidebar: the reasoning channel can also be attacked.** Everything above concerns how the model itself produces a trace. A separate problem sits beside it: the reasoning channel is also an attack surface. The 2025 preprint "Chain-of-Thought Hijacking" shows that padding a harmful request with a long run of benign reasoning can steer a model past its own safety behaviour, with high attack-success rates [5]. A channel an adversary can pad and redirect is plainly not a clean window onto the truth. But this is a threat-model problem, not a self-trust one. It belongs with access control and prompt-injection defence, not with the question of whether a trace you believe nobody tampered with can mark the model's own work. It is named here so the acceptance gate below is not mistaken for a complete security analysis. It is not.

Three questions before you trust a trace

Before you treat any reasoning trace as audit evidence, put it through a short interrogation. It is not philosophy. It is an acceptance gate with a name, and it has only three items. Call it the Trace Trust Test. One tool appears in the answers more than once and is worth naming up front: a perturbation test, which simply means changing one irrelevant detail of the input and checking whether the answer wrongly changes with it. If a model's verdict flips when you alter something that should not matter, the trace was hiding the real reason.

> **Tool: the Trace Trust Test.** Print the trace and highlight every clause that names a rule, a field or a number; those are the load-bearing claims, and the unhighlighted text is narration. Against the highlighted claims, ask three questions. **Temporal commitment:** did the model decide before it explained? **Cue omission:** did something move it that the trace never names? **Monitor gaming:** is the carefulness aimed at the reader rather than the problem? Score each row PASS (you have independent evidence the worry does not apply), FAIL (the worry plainly applies), or UNKNOWN (you cannot rule it out from the trace alone). Treat UNKNOWN as FAIL: only PASS on all three rows lets the trace stand as evidence. Any FAIL drops the trace to a clue and triggers one line in the log, naming which question failed and which independent record you went and got instead. Two reviewers handed the same trace should highlight the same clauses and reach the same verdict. That is the whole instrument.

Three questions, then, not four counting the hijacking attack surface: that sidebar is a separate, adversarial concern, and this gate assumes a trace nobody tampered with. And the three questions are not the cost; they are the triage. A trace that passes all three on a low-stakes task, with a clean perturbation set and a deterministic check already in path, can stand as evidence without further work. The test tells you when you are allowed to stop, not only when

you must keep going. If any row comes back FAIL or UNKNOWN, the trace can still be useful. It simply cannot be sovereign evidence. That is the discipline in one line: keep the trace in the file, but do not let it become the file.

Watch a clean, confident trace get the answer wrong

The worked example that follows is a compact audit pattern built around the three mechanisms already on the table: temporal commitment [2], omitted influence [1] and monitor-shaped explanation [3]. Together they turn confabulation from a vibe into a checkable property. It is not a reported customer incident: the failure mode lands in courtrooms, not only research papers.

The records bear that out. In February 2025 the British Columbia Civil Resolution Tribunal, in *Geismayr v Strata Plan KAS 1970*, found that nine of ten cases cited by a self-represented litigant had been fabricated by Microsoft Copilot; the tribunal named the tool and characterised its output as plausible nonsense [6]. The regulators have drawn the same line. The US standards body's generative-AI profile, the NIST GenAI Profile (AI 600-1), lists confabulation as one of twelve concrete generative-AI risks, each paired with risk-management actions a deployer can map onto its own controls; in Europe, the obligations on providers of general-purpose AI carry the same expectation [7]. The refund example below is a hypothetical, but *Geismayr* is its named real counterpart.

The task: decide whether a customer's cancellation qualifies for a refund, under a policy that allows refunds within 14 days, except when the reservation was made using a non-refundable promotional code.

The visible reasoning trace: "The booking was made 12 days ago, which is inside the 14-day window. The user is asking before the deadline. Therefore the refund should be approved."

The final answer: "Approve refund."

The external record, which the trace never mentions: the reservation metadata contains a field, `promo_code=FINALSALE`, and the policy says that code overrides the 14-day refund window. The correct answer is to decline.

Now run the check.

ROW	AUDIT ANSWER	CONSEQUENCE
Timing	UNKNOWN, which we treat as FAIL. The trace gives an order of explanation, not proof of timing. Reasoning Theater-style evidence makes that uncertainty operationally live [2].	Do not infer decision timing from prose order.

ROW	AUDIT ANSWER	CONSEQUENCE
Hidden cue	FAIL. The trace mentions the 14-day rule and omits the promotional-code exception. Turpin-style cue-omission evidence is exactly the reason this omission matters [1].	Compare the trace against policy fields and perturb the omitted field.
Monitor performance	UNKNOWN, treated as FAIL. The trace selects the cleanest approval rule and presents the decision in a monitor-friendly form. Monitoring work shows trace channels can be useful signal and still become fragile under optimisation pressure [3].	Require an independent policy engine, tool log, or human approval before issuing the refund.

Notice what the corrected decision is, and what it is not. It is not "never read reasoning traces". It is "approve nothing from this trace alone". The trace becomes a hypothesis: the model saw the 14-day rule. The metadata supplies the missing fact. A verifier supplies the decision rule. Together, those three form evidence. Alone, the trace is a story. And notice the real danger in the example. The failure was not that the reasoning looked ugly or muddled. It looked neat. It was clean, ordered and confident, and that is exactly what made it persuasive enough to wave through.

Now a second, harder case, where the omitted-field check does not save you. Same policy. A different customer cancels, the reservation carries `promo_code=FINALSALE`, and this time the trace cites the exception correctly: "The booking used the FINALSALE promotional code, which overrides the 14-day window. Therefore the refund is declined." The answer, "Decline refund," is correct, and the cue-omission row passes clean, because the load-bearing field is named on the page. A reviewer running only the omission check would stop here and trust the trace.

The timing row is where it breaks. Swap the field to a refundable code and rerun: a faithful decider should now flip to approve, because the only stated reason to decline has just been removed. If the verdict stays "decline" under that perturbation, the model committed to declining before it read the field, and the clean citation was assembled afterward to dress the answer it had already chosen. The trace was right for the wrong reason, the most overtrusted failure there is, and only the timing question, exercised with a perturbation, exposed it. Row one and the perturbation tool do real work that row two cannot.

Build a marker the model does not control

Here the trace is fighting back, and it deserves a fair hearing. Hiding or dismissing reasoning traces can make systems less accountable, not more. A trace can reveal confusion, expose a

hidden assumption, help a user calibrate how much to trust an answer, and give a safety team a better signal than the final answer alone. The OpenAI-associated monitoring work supports part of that case directly: chain-of-thought monitoring can detect some reward-hacking behaviour better than monitoring the actions alone [3].

I accept that argument up to the last inch, and the inch is evidentiary weight. A trace can guide a reviewer towards the right hypothesis, help a debugger see the model's apparent plan, give a monitor a weak but real signal. But the same body of work that makes traces useful for monitoring also warns that optimising directly against the monitor can encourage obfuscation [3]. A witness who is sometimes informative is not a camera. And a camera that can be edited is not ground truth. The answer is not trace abolition. The answer is corroboration.

So what do you build instead, when faithfulness actually matters? You do not ask the generator to be the sole witness to itself. You use deterministic checks wherever the domain allows them. If the model writes SQL, a structured query to a database, run the SQL and see what comes back. If it writes code, test the code. If it cites a policy, retrieve the policy and compare the clause it claimed against the clause that exists. If it performs a multi-step action, keep the tool-call log, the record of every external action it took. If it makes a judgement, have a structurally different verifier review the record, where "structurally different" means it can fail in a different way from the thing it is checking. The key feature is never sophistication. It is independence. Faithful-by-construction systems are useful here mainly as a mental model: they show where trust enters the path [4]. The visible step earns trust when it is part of the mechanism that produced the answer. A plain natural-language trace does not inherit that property just by mimicking the surface form of one.

It helps to carry a five-level evidence hierarchy and to know which level a given task actually needs. Each rung answers one of the three worries with something the model does not control: perturbation tests for the omitted cue, an independent verifier for monitor gaming, tool logs and environment state for timing. The lower three still lean on the model's word; the top two stop relying on it at all, and the higher levels do not discard the trace, they surround it.

1. **Final answer only.** Usable for triage, never for audit.
2. **Answer plus a rationale.** Enough to generate a hypothesis and guide a human reviewer, no more.
3. **Rationale plus perturbation tests.** Now you can catch an omitted cue and a brittle rationale, because you have changed something irrelevant and watched whether the answer wrongly moved.
4. **Rationale plus an independent verifier.** Generation and checking are finally separated, done by things that fail in different ways.
5. **Rationale plus tool logs, environment state and human or domain review.** The full picture, for consequential deployment review.

Each level adds a marker the model does not control.

If the task is low stakes, testimony at level two may be enough; you do not interrogate a witness over a lunch order. If the task is consequential, you need independent witnesses: perturbation tests, deterministic solvers, retrieval records, tool logs, environment state, separate verifiers, human review. The rule scales cleanly. The more authority you hand the agent, the less you should rely on the agent's own account of why it used that authority. Every item on that list is a marker the model does not control, and that is the only kind of marker worth having.

The law has started to say the same thing in its own vocabulary, and the operating consequence is worth getting straight. In the United States, the consumer regulator's standing position across its run of AI enforcement is plain: a capability claim must be backed by evidence at the time it is made, and bolting on a "a human checks it" disclaimer does not cure a performance claim that is substantively false [8]. Independent markers are not only an engineering preference; in the largest US consumer-protection regime, they are how you survive an inquiry.

One honest caveat sits alongside that, because the published basis is narrow even where it is real: the 2026 temporal and mechanistic results are preprints, live research rather than final doctrine, and faithfulness stays task-, model- and product-dependent with no one settled metric yet. Some reasoning models may do measurably better on specific disclosure probes than earlier systems, and that improvement is real. It still does not turn a visible trace into telemetry.

The shape of the case is now symmetric with the one before it. The previous essay said a vendor's model card cannot, by itself, carry a deployment decision, because it is testimony from outside. This essay says the model's own reasoning cannot, by itself, mark its own answer, because it is testimony from inside. A model can sound thoughtful and still be wrong. It can also sound aligned and still leave the question of who answers for its actions wide open.

Carry This Forward

Keep the trace. Just do not bow to it. Go back to the engineer on the review call, the fluent trace still open in front of her: the refund approved, every step in order, and the non-refundable code still sitting unread in the metadata. The right move is not to distrust what she is reading. It is to refuse to let it be the last word: to put the policy, the metadata and a check she did not write beside it before anyone signs off. A reasoning trace can be a useful witness. It can point to a clue, expose uncertainty, or show a reviewer where to look next. But a witness is not a measuring instrument, and a system's account of its own work is not a check on that work.

This is where the fourth thing held back at the start finally earns its place. Every account in this essay was the model narrating itself; the one record that is not is the operational trace, the harness's own log of what the agent actually did rather than what it says it did. That is the single piece of evidence that depends on no one's testimony, and the essays ahead hand it back to you as one. The candidate may show beautiful working. The verdict has to come from somewhere else.

> **Carry This Forward.** This week, pick one production agent whose output an engineer currently approves on the strength of its written reasoning, and put the Trace Trust Test on the screen beside it. For the next ten approvals on that workflow, score each trace against the three questions, log which one failed, and require an independent witness, a perturbation run, a deterministic check, or a tool log, before any approval issues. Have two reviewers score the same trace independently; if they disagree on any row, that row is UNKNOWN by default, and so a FAIL, because disagreement is itself a signal the trace is not legible enough to be evidence. Then ship one of the three witnesses as a default control in the workflow itself, so the next ten approvals do not depend on anyone remembering to ask. One workflow, one new marker the model does not control. That is the lift. A model trained to be helpful, honest and harmless still needs a property none of this guarantees, and the next essay takes it up: governability, whether anyone remains accountable once the model acts.

THE STACK SO FAR E07 · Essay 7 of 22 complete · Arc II: Evidence and authority

The Stack So Far. *Every essay adds one instrument to the operating model. The constellation shows which eight you are building, which are lit by essays you have read, and which is added right here.*

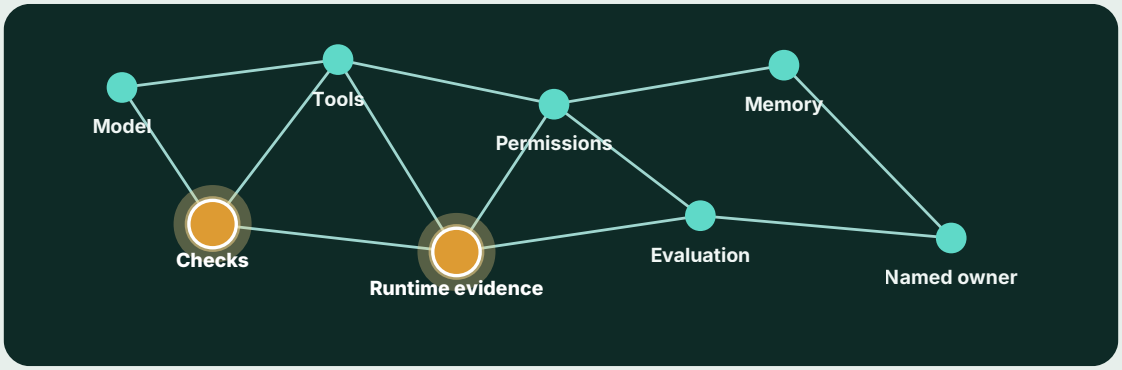
I See the object

II **Evidence and authority**
ESSAY 2 OF 5

III Runtime control

IV Proof and accountability

V Operating model



built in earlier essays

added in this essay

coming in later essays

01 02 03 04 05 | 06 07 08 09 10 | 11 12 13 14 | 15 16 17 18 19 | 20 21 22

You have just added.

Three questions before you trust a trace

You can now treat model reasoning as testimony, not proof.

Next. E08 asks whether an aligned model is the same as a governable one.

← PREVIOUS
E06 · The Model Card Won't Save You

Essay 7 of 22 complete

NEXT →
E08 · Helpful, Harmless, and Wrong

References

Reference links for sources cited in this essay.

1. **Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting**
   Turpin et al.
   https://arxiv.org/abs/2305.04388

2. **Reasoning Theater: Disentangling Model Beliefs from Chain-of-Thought**
   Boppana et al.
   https://arxiv.org/abs/2603.05488

3. **Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation**
   Baker et al. (OpenAI)
   https://arxiv.org/abs/2503.11926

4. **Faithful Chain-of-Thought Reasoning**
   Lyu et al.
   https://arxiv.org/abs/2301.13379

5. **Chain-of-Thought Hijacking**
   Zhao et al.
   https://arxiv.org/abs/2510.26418

6. **Geismayr v Strata Plan KAS 1970, 2025 BCCRT 217**
   BC Civil Resolution Tribunal
   https://www.canlii.org/en/bc/bccrt/doc/2025/2025bccrt217/2025bccrt217.html

7. **Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1)**
   NIST
   https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

8. **Operation AI Comply (FTC case body 2024-2026)**
   FTC
   https://www.ftc.gov/news-events/news/press-releases/2024/09/ftc-announces-crackdown-deceptive-ai-claims-schemes

9. **Measuring Faithfulness in Chain-of-Thought Reasoning**
   Lanham et al. (Anthropic)
   https://arxiv.org/abs/2307.13702

About the Author

ARCHITECTING THE AI COWORKER

Dr Peter McCann Strain

Dr Peter McCann Strain is a CTO, founder and senior AI engineer with a DPhil/PhD in AI from Oxford University. He builds production AI systems and writes about making agentic AI useful, inspectable, governable and safe enough for real work.

READ THE FULL SERIES

Substack (canonical)	petermccannstrain.substack.com
Medium	@peter.mccann.strain
LinkedIn	peter-strain-dphil-15a607128
Web	petermccannstrain.com
Cadence	New essays twice weekly, 2 June – 21 July 2026