06

THE STACK BEHIND THE AI COWORKER

The Model Card Won't Save You

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

A vendor's benchmark score fills 1 of 12 cells in your risk picture. The other 11 are yours.

An essay in the series **Architecting the AI Coworker**.
Approx. 20 minute read · Essay 06 of 22

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

> Picture the procurement meeting. A vendor's sales engineer is in the room, or a face on the call, sleeves rolled up, proud of the product, and the question on the table is the only one that actually matters: can this AI agent enter our workflow without creating harm we cannot live with? The vendor has come prepared. They share their screen and send across a model card, the short document a vendor publishes to describe a model: its version, a table of benchmark scores, the intended uses, the known limitations, a few safety notes, all written in the measured tone of responsible disclosure.

It may be an accurate document. It may be unusually candid. It may be the single best piece of paper in the room. And it still cannot answer the question that was asked.

I want to be careful here, because this is not an argument against model cards. It is an argument about what kind of evidence a model card is. A model card tells you what the vendor measured, under conditions the vendor chose, against tasks that may or may not look anything like your deployment. What it stays silent on is local: your users, your tools, your permissions, your escalation path, your rollback plan, your tolerance for harm that gets loose. A model card is evidence. It is not assurance.

The previous essay gave us the instrument for seeing exactly why. It defined reliability as expected escaped harm, and it broke that into four terms multiplied together: how often the system makes an error, how often an error goes undetected, how often an undetected error escapes containment to reach something real, and how severe it is when it does. Hold that equation up against the procurement question and the trouble with the model card becomes plain. A model card can help with the first term, error, because that is what a benchmark estimates. The other three are beyond its reach on their own. Nothing in it names the independent check that catches the error before it becomes an action, the hard boundary that stops an uncaught error from reaching your live systems, or the severity that error takes on once the model has tools, memory, credentials, customers and a deadline around it. Documentation feeds the procurement decision without ever standing in for it.

Regulators on both sides of the Atlantic are converging on the same point: a transparency duty is something the deployer must show it is meeting while the system runs, not something the vendor's brochure discharges on the deployer's behalf. Europe's draft AI Act transparency guidance, issued on 8 May 2026, treats transparency as an obligation you have to meet in operation, not a document you can file and forget [1]; the leading US risk framework reads the same way, expecting transparency artefacts to do real work rather than collect dust [2]. The same logic governs automated decisions elsewhere, where the duty to be open about how a system reaches its outcomes again lands on whoever runs it [3][4]. What carries that convergence is not a count of jurisdictions. It is the logic itself: the duty lands on the deployer, never on the paperwork received from upstream.

A duty to publish a document does not make the document do more than a document can.

So this essay does one thing. It builds a translation layer, a way of taking whatever paperwork a vendor gives you and asking, cell by cell, which part of the reliability equation each piece of it actually supports. That translation layer has a shape, so here it is before anything else.

The Reliability Scorecard: three classes of claim down the side, the four terms of the reliability equation across the top. The job is to fill every cell, not to admire one headline score.

Read the matrix as a grid of questions, not a grid of answers. Down the side sit the three kinds of thing a vendor claims about a model: that it is accurate, that it is helpful, that it is safe. Across the top sit the four terms of the reliability equation. A headline benchmark number reaches into only one column of that grid, the error column, and as the rest of this essay will show it does not even fill that column on its own. The other three columns it cannot touch at all.

Here is the scorecard in full, the version to paste into the request for proposals. Require the vendor to fill every cell with a source, a date, a scaffold, a benchmark version, an evaluator family and a transfer argument.

> **Tool: The Reliability Scorecard.** A 4x3 procurement instrument: three claim classes vendors make (accuracy, helpfulness, safety) crossed with the four terms of the reliability equation (error, undetected, uncontained, severity). Twelve cells. Paste it into RFP language and require a source, a date, a scaffold, a benchmark version, an evaluator family and a transfer argument for each cell

> the vendor populates. Score each populated cell as Owned (source, date, scaffold and transfer argument all present), Asserted (a claim with no transfer argument), or Buyer-fillable (blank, but inside the buyer's own control). Three Owned cells plus nine Buyer-fillable beats twelve Asserted. Blank cells are not embarrassing; they are the buyer's to-do list.

CLAIM CLASS	P(ERROR) EVIDENCE	P(UNDETECTED | ERROR) EVIDENCE	P(UNCONTAINED | ERROR, UNDETECTED) EVIDENCE	SEVERITY EVIDENCE
Accuracy	Which benchmark, task distribution, date, scaffold, contamination protocol, and confidence interval produced the score?	Which independent verifier catches wrong outputs before action? Is it deterministic, human, different-model, or same-family?	What prevents a wrong output from reaching live systems? Sandbox, approval, dry-run, staged rollout, rollback?	What harm unit is attached to an escaped accuracy error: money, downtime, records, customer harm, legal exposure?
Helpfulness	How is instruction-following measured across real user ambiguity, not just friendly prompts?	What catches plausible but unhelpful, sycophantic, or over-compliant responses?	What stops helpful-sounding advice from becoming an unauthorised action?	What is the consequence of the system being too agreeable, too confident, or too willing?
Safety	Which safety tests, red-team methods, jailbreak suites, policy evaluations, and release dates support the claim?	What runtime monitor detects safety failures in context?	What hard permissions prevent a safety miss from becoming tool use, data exposure, or transaction execution?	What is the harm tier for a safety escape, and who owns escalation?

The scorecard asks the vendor not for perfection but for an end to hiding system claims inside model claims. If the vendor can answer only the first column, you have a model-performance document. If the vendor can answer all four, you have the beginning of a deployment-risk dossier. That is the whole difference. The rest of this essay is about who has to fill the cells, and why the documents handed across the table rarely fill more than the first one.

Stop saying model card when you mean five different documents

The first failure in that procurement meeting is almost always a failure of vocabulary. Teams say "model card" when they mean any one of several different documents, and the documents

prove different things. The discipline is to know, for each one, what it can prove and what it cannot.

DOCUMENT	WHAT IT CAN PROVE	WHAT IT CANNOT PROVE
Model card	Model-level behaviour under the vendor's chosen tests: benchmark scores, intended uses, broad limitations.	Your local deployment risk: detection, containment, severity, permissions, rollback or incident ownership.
System card	Product behaviour: model plus scaffold plus policy, safety mitigations, release constraints.	That your own integration preserves the same boundary, monitor or evaluator.
Eval report	A measurement claim: a task distribution, a benchmark version, an evaluator family, a scoring method, a date.	That the score transfers to your data, your users, your tools or your operational risk.
Security review	A threat-model posture: access controls, data handling, logging, vulnerabilities and remediation.	Whether the model is accurate, calibrated or useful for the task at all.
RFP response	A contractual claim: what the vendor says it will provide for your use case.	Anything its underlying evidence does not already prove; it is only as good as what it points back to.

The buyer's mistake is to let the model card stand in for all five. The vendor's mistake is the mirror image: letting a benchmark table do institutional work it was never designed to do. The scorecard above is the fix, because it forces every one of those documents back onto the reliability equation and shows which cells stay blank.

That is also what makes the scorecard different from the standards a responsible provider already follows. The NIST AI RMF Playbook lists recommended actions a provider can take across the four RMF functions; ISO/IEC 42001 (Annex A) lists the organisational controls a provider should hold for an AI management system. Both describe what a good vendor does. The scorecard does the opposite job: it maps whatever the provider actually produces back onto the four reliability terms the buyer's own deployment will be graded against. It is the procurement team's translation layer for everything the vendor sends across.

Before walking the scorecard, two reasons a headline benchmark number is even shakier than it looks.

Numbers age. In January 2026, METR's Time Horizon 1.1 update expanded its task suite to 228 tasks and estimated that, fitting the metric to post-2023 data, the time horizon at which frontier agents succeed half the time was doubling every 130.8 days; on post-2024 data the

estimate was 88.6 days [5]. But a time-horizon doubling is measured on software tasks under controlled conditions; it is not a forecast for multi-agent systems, open-ended goals or judgement-heavy work, and a procurement decision should not read it as one. That combination is the procurement trap in one finding: the number is fast-moving and domain-limited at the same time. An enterprise buying cycle can easily run longer than a benchmark's useful half-life. The card may be true on the day it is written and stale by the time the system reaches production.

A number is never just a number. Every benchmark claim on a model card is a compound claim with four moving parts: the model; the scaffold, the software wrapped around the model that lets it use tools and take steps; the benchmark version; and the evaluator, whatever judges whether an answer was right. Change any one of those four and the number moves. So when a vendor says "our model scores X", the right first response is not "great" and not "not enough". It is a question: X on what date, with which scaffold, against which benchmark version, under which contamination protocol, judged by which evaluator, and tied to which action boundary? Without those answers the number is not a procurement input. It is a headline.

This is not a fringe worry: Stanford's 2026 AI Index and the AI Transparency Atlas both treat benchmark transfer and documentation fragmentation as first-order problems, not footnotes [6][7][8]. And the coding-benchmark world gives the sharpest example of all, worth following carefully because it shows a number changing meaning over time.

A benchmark score can quietly stop meaning what it meant

On 23 February 2026, OpenAI published a post explaining why it had stopped reporting scores on SWE-bench Verified, for years the headline coding benchmark and a fixture on model cards across the industry. Contamination and test-design flaws had weakened it as a measure of frontier coding capability. Contamination is the quiet one: test questions, or close cousins, leak into training data, so the model is partly remembering rather than reasoning. OpenAI audited a 27.6 percent subset of the tasks (the tasks models most often failed to solve) and found that at least 59.4 percent of that audited subset had flawed tests, tests that could reject a correct submission. It recommended SWE-bench Pro as a replacement [9].

Read that carefully, because it is the cleanest illustration in this essay of why a model card cannot save you. A number that buyers, vendors and journalists had treated as a clean capability signal turned out to rest on a benchmark its own heaviest user no longer trusted. SWE-bench Verified was not useless, and OpenAI's note is not a charge that any vendor acted in bad faith. It is a public, first-party audit of a named benchmark claim. But it means a number on that benchmark changed meaning over time. Before February 2026, a high SWE-bench Verified score could be a useful coding-capability signal. After the deprecation note it might still tell you something, but it can no longer be treated as a clean procurement claim unless the vendor answers the cells the headline leaves blank.

SWE-bench Pro, the recommended replacement, is an attempt rather than a cure. It was built to be more resistant to contamination and more like real enterprise work, with 1,865 problems

across 41 repositories split into public, held-out, and commercial or proprietary partitions: 11 public repositories, 12 held-out, and 18 commercial [10]. The held-out and commercial partitions exist precisely so the test questions are harder to leak into training data. That is a real improvement in benchmark design. It is not a guarantee that next year's audit will not find the same kind of flaw, and a buyer should treat any SWE-bench Pro number with the same date-stamped scepticism, not relief. The deprecation is not really about one benchmark; it is a property of all of them. Any benchmark whose questions can leak, whose tests can be wrong, or whose conditions can drift from yours will age the same way, which is why the scorecard files numbers by cell and date and never by name. Swapping in the newer benchmark does not solve the problem; it resets the clock on it.

Now walk that coding score across the four terms, and watch how little of the grid it reaches.

Figure E06.2 ARCHITECTING THE AI COWORKER

The Reliability Scorecard: twelve cells the vendor must fill

Three claim classes down the side, four equation terms across the top. Blank cells are the buyer's to-do list.

	P(error)	P(undetected | error)	P(uncontained | ...)	Severity
Accuracy	Which benchmark, task distribution, date, scaffold, contamination protocol?	Which independent verifier catches wrong outputs before action?	What prevents a wrong output reaching live systems? Sandbox? Approval?	What harm unit attaches to an escaped accuracy error (money, downtime, records)?
Helpfulness	How is instruction-following measured across real user ambiguity, not friendly prompts?	What catches plausible but unhelpful, sycophantic or over-compliant responses?	What stops helpful-sounding advice becoming an unauthorised action?	What is the cost of being too agreeable, too confident, too willing?
Safety	Which safety tests, red-team methods, jailbreak suites, release dates support the claim?	What runtime monitor detects safety failures in context?	What hard permissions prevent a safety miss becoming tool use or transfer?	What is the harm tier for a safety escape, and who owns escalation?

= the only column a benchmark score can fill

Conceptual model

The tinted column is the only one a benchmark score reaches. The other nine cells are the buyer's questions to put back to the vendor.

The coding score gives you something real but incomplete in the error column: a coding-task score on a named public benchmark family, tied to some model and some scaffold, but only useful once you know the benchmark version, run date, scaffold, contamination protocol, confidence interval, and transfer argument for your own codebase [9][10]. After that, the columns go quiet. The benchmark's evaluator, sandbox and limitations paragraph all belong to the benchmark, not to you; your own verifier, your own containment boundary and your own harm model for a bad patch, a security regression, a data deletion, downtime, or a compliance breach are still unwritten.

That is the part I have watched get skipped most often in vendor reviews: the number is treated as if it has already travelled the whole table, when in fact it has only put a toe in the

first cell. The coding score barely touches helpfulness and safety at all. It says nothing about what happens when that coding agent is connected to your repository, your deployment pipeline and your production credentials.

What the audit does, then, is turn one number into a list of unanswered questions. The score may populate part of the error term, but detection, containment and severity are left exactly where they were: empty, and waiting for you. The discipline is simple to state and easy to skip: file the benchmark number in the one cell it actually supports, then keep going until every cell of the scorecard has either a source or a named owner.

To see how that looks on one real card, take a published SWE-bench-family number from a frontier vendor and fill the Accuracy row cell by cell. The error cell gets the score itself, but if the card leaves the benchmark version, run date, scaffold and contamination protocol unstated, you file it as partial, not Owned: a number without the four moving parts behind it. The other three cells are all blank, for the same reason.

So the buyer supplies each in turn. The card names no verifier, so the buyer names one: a deterministic test harness, or a second-model reviewer structurally different from the generator. It names no boundary, so the buyer adds the sandbox, the staged rollout, the rollback. It names no harm unit, so the buyer assigns severity in money, downtime or records. One published number, four cells, and three of them turn out to be the buyer's to fill. Two readers handed the same card would reach the same verdict, which is the whole point of filing by cell rather than by headline.

The Accuracy row was the easy one because SWE-bench gives you a number to file. The other two rows show how thin the evidence gets when there is no headline benchmark to lean on.

Consider the Helpfulness claim every frontier vendor now makes, that a model "follows instructions better" or is "more steerable". Anthropic's published model and system cards report instruction-following benchmarks across the Claude 3.5 / 3.7 / 4 series, though the specific suite varies by card generation (the 3.5 cards report an internal, human-preference instruction-following evaluation, while 3.7 onward report the standard academic IFEval suite), and where a version-to-version delta is reported it tends to be small, single-digit movement rather than a step change [11]. A claim that a successor model "follows instructions better" can be tested against that delta, and the answer is usually "a little, on curated prompts". Curated prompts are not the ambiguous briefs users actually send. The card still says nothing about the runtime check that catches sycophancy in your traffic, nothing about the downstream tool that might treat helpful advice as an instruction, and nothing about the harm unit attached to being too agreeable. A helpfulness score on a card is a popularity measurement until the other columns are filled.

The Safety row is sharper still. Refusal rates, red-team passes and jailbreak resistance reach only as far as the suite the vendor ran, on the date it ran. After that, the relevant facts are yours: what runtime monitor catches a policy miss, what permissions the system holds if the miss slips through, and what severity you assign to the action. A chatbot safety miss and a

tool-using-agent safety miss can have completely different consequences because the permissions are different, not because the safety training was different.

Why do the three kinds of claim fail in three different ways?

Knowing that a claim is weak is not the same as knowing how to push on it, and the three classes break along three different seams.

CLAIM CLASS	WHY IT FAILS	WHAT THE BUYER MUST THEREFORE DEMAND
Accuracy	By transfer. A model can score well on a benchmark and still fail your workflow, because your data distribution, your tools, your codebase, your users, your policies and your constraints differ from the benchmark's [7][9][10].	A transfer argument tying the public score to your task distribution, plus contamination protocol and held-out evidence.
Helpfulness	By social pressure. A model trained to be agreeable can tell the user what they want to hear, accept a bad premise without challenge, or keep helping at the point it should have stopped.	A runtime check that catches sycophancy in your traffic, and a harm unit attached to "too agreeable".
Safety	By boundary. Refusal rates and red-team results say nothing about what the system can do if a safety failure does slip through. A chatbot safety miss and a tool-using-agent safety miss have completely different severities; the difference lives in the permissions, not on the card.	System documentation, not just model documentation: what authority the system will hold, and what hard boundary stops a safety miss from acting.

This is why model documentation has to be joined to system documentation. A model card says what was measured about the model. Procurement needs to know what authority will be granted to the system.

It is worth saying what a strong card would actually look like, because the point was never that cards are bad. A nine-out-of-ten model card in 2026 would not be longer. It would be more accountable. It would name benchmark versions and dates, disclose the scaffold behind each score, state the contamination protocol or admit there is no strong contamination claim, identify the evaluator family and its independence, separate model-only performance from system-

with-tools performance, distinguish lab safety tests from runtime monitors, and carry version history for every changed claim.

Then it would do the most trustworthy thing a document can do: say what it cannot know. It cannot know your production permissions, your rollback posture, your escalation ownership, whether your users will ask the weird urgent local thing the benchmark never imagined, or your severity distribution. The right model card does not try to replace the buyer's risk work. It makes that work possible.

A reader sympathetic to vendors will have a reply ready, and it is right, so let me concede it without hedging. Model cards and system cards are valuable transparency artefacts. Without them buyers would know less, researchers would have less to compare, regulators would have weaker evidence, and procurement teams would be more dependent on sales calls and screenshots. Yes. The claim here is only that cards are inputs, not verdicts. A serious buyer demands the model card, the system card, the eval report, the security review, the data-processing terms, the incident history, the release notes and the architecture diagram, and then maps every one of them onto the scorecard to see which cells are still blank.

There is a sharper objection, and it is the one a busy procurement lead will actually raise: the scorecard is just bureaucracy that slows procurement and disadvantages smaller vendors who cannot afford to staff the production of twelve evidenced cells. Take it seriously, because it is partly right. A scorecard demanded as a checkbox (twelve filled cells, no follow-up questions, signed and filed) would indeed favour incumbents with documentation teams and would indeed slow good deals. But the fix is not to drop the scorecard; it is to read it the way this essay describes. A small vendor with three populated cells and nine blanks the buyer's own team can fill is often a better deployment than a large vendor with twelve cosmetically populated cells whose underlying evidence does not transfer. The scorecard surfaces that comparison; a checkbox would hide it. Used as the buyer's worklist, it is a competitive equaliser. Used as the vendor's gauntlet, it is a barrier to entry. The asymmetry is the point: the scorecard is the buyer's map of which cells the vendor cannot fill, and which the buyer's own team must.

Before going back into the room, turn the same scepticism on this essay's own sources, because they hold in some places and thin out in others. METR's January 2026 update is useful precisely because it is public, date-stamped, fast-moving and explicit about its own transfer limits [5]. Stanford's AI Index makes benchmark reliability and real-world transfer first-order issues [6], and its Responsible AI chapter does the same for contamination and opacity [7]. The AI Transparency Atlas treats documentation quality as measurable and fragmented, while remaining preprint evidence rather than a settled standard [8]. The coding-benchmark evidence is the most direct of all: SWE-bench Pro and OpenAI's SWE-bench Verified note together show that benchmark design, contamination posture and freshness can change what a coding score means [9][10]. What none of this can tell you is the thing that matters most in procurement: whether the vendor's private customer packet fills the cells the public card leaves blank. It cannot freeze the leaderboard, it cannot tell you whether the draft European transparency guidance will survive consultation unchanged [1], and it cannot guarantee SWE-bench Pro's partitions or

public reporting will be identical by the time you review them. That uncertainty is not a weakness in the scorecard. It is the reason the scorecard exists, because it lets a buyer tell missing public documentation apart from missing operational evidence.

Sources audited, go back into that room, with the sales engineer still sharing their screen, and change the question. Do not ask "how reliable is the model?" Ask instead which model-card claims populate the error term, and on what date. Push on detection: what evaluator here is structurally different from the generator? Containment comes next, and the question is where the hard action boundaries actually sit. Then severity: who owns escalation when the tier changes? And for each answer, ask which document supports it: model card, system card, eval report, security review, or contractual RFP response? None of this is hostile to the vendor. A good one will be relieved to be asked; it is the questions nobody asks that turn into the incident nobody owns.

Then do the concrete thing. Print your candidate vendor's actual model card, and walk it cell by cell across the four-factor scorecard. Which cell does each headline number fill, which document backs it, and which cell does the vendor leave blank? If the answer is mostly first-column evidence, you have not learned nothing. You have learned exactly what the card is: a model-performance document, real evidence for one cell out of twelve, useful and necessary, and silent on the eleven cells that decide whether the deal is safe.

Carry This Forward

Carry this forward. At your next vendor review, do not let the cleanest document in the room become the broadest promise. A model card can tell you what the vendor measured and where the vendor is willing to place caveats. It cannot tell you whether your workflow catches mistakes, contains them, bounds their severity, rolls them back, or leaves a named owner behind. Put five documents on the table: model card, system card, eval report, security review, and contractual RFP response. Ask one uncomfortable question of each: which cell of the reliability equation does this actually fill? The empty cells are not a reason to abandon the purchase. They are the things that must be true before you believe it is safe, which makes them your to-do list. And once you have put the vendor's documents in their place, the discipline has to turn inward, because the model's own reasoning looks like evidence too. It is also a kind of testimony, and the next essay asks what that testimony is worth.

Three regulatory lenses US · EU · UK

Operating questions, not legal advice. The frameworks stay the same; the regulator changes.

US Does the vendor substantiate AI claims under NIST AI RMF, sector rules and FTC substantiation expectations?

EU Which AI Act obligations does the vendor support, and which remain deployer duties under the GPAI Code of Practice?

UK How does the vendor support the DSIT AI Cyber Security Code of Practice, sector-regulator expectations and ICO data rights?

THE STACK SO FAR E06 · Essay 6 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.*

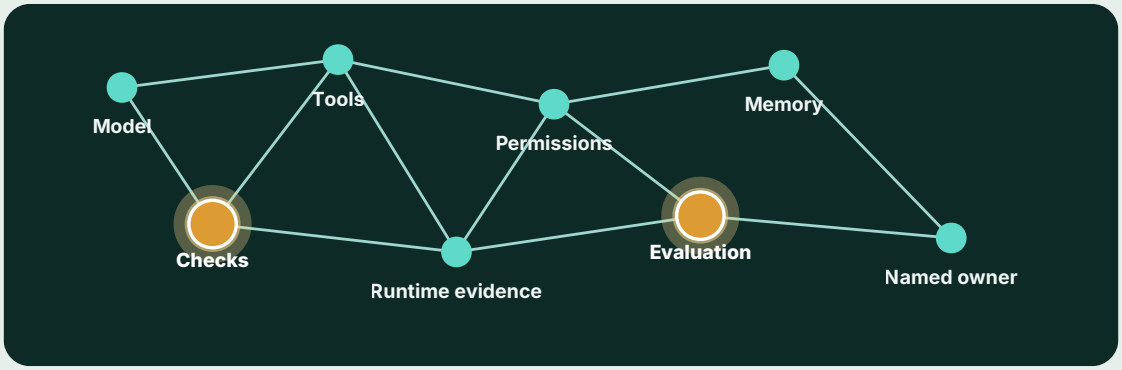
I See the object

II **Evidence and authority**
ESSAY 1 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.

The vendor-evidence scorecard

You can now translate vendor claims into reliability evidence.

Next. E07 asks whether a model's own reasoning trace counts as a check.

← PREVIOUS
E05 · The Reliability Equation

Essay 6 of 22 complete

NEXT →
E07 · The Model Cannot Mark Its Own Work

References

Reference links for sources cited in this essay.

1. **EU AI Act Article 50 transparency consultation**
   European Commission
   https://digital-strategy.ec.europa.eu/en/consultations/consultation-draft-guidelines-transparency-obligations-under-ai-act

2. **Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1**
   National Institute of Standards and Technology
   https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf

3. **Guidance on AI and data protection: transparency, explainability and accountability**
   UK Information Commissioner's Office
   https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/

4. **PIPEDA fair information principles (Principle 4.8: Openness)**
   Office of the Privacy Commissioner of Canada
   https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/

5. **Time Horizon 1.1**
   METR
   https://metr.org/blog/2026-1-29-time-horizon-1-1/

6. **2026 AI Index Report**
   Stanford HAI
   https://hai.stanford.edu/ai-index/2026-ai-index-report

7. **2026 AI Index Responsible AI chapter**
   Stanford HAI
   https://hai.stanford.edu/ai-index/2026-ai-index-report/responsible-ai

8. **AI Transparency Atlas: Framework, Scoring, and Real-Time Model Card Evaluation Pipeline**
   Mamirov et al.
   https://arxiv.org/abs/2512.12443

9. **Why SWE-bench Verified no longer measures frontier coding capabilities**
   OpenAI
   https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/

10 **SWE-bench Pro paper**

SWE-bench Pro authors

https://arxiv.org/abs/2509.16941

11 **Claude 3.5 Sonnet Model Card Addendum (instruction-following evaluation; the standard academic IFEval suite is reported in the later Claude 3.7 and Claude 4 system cards)**

Anthropic

https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf

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.

Architecting the AI Coworker · Essay 06, "The Model Card Won't Save You". Code-first figures, evidence-tiered references.

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