01

THE STACK BEHIND THE AI COWORKER

There Is No "It"

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

Stop saying "the AI did it." Rewrite it in six named fields, and the controls that would have stopped it appear as blanks.

An essay in the series **Architecting the AI Coworker**.
Approx. 12 minute read · Essay 01 of 22

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

> "The AI did it." Four words, a small shrug, and a room full of capable people relaxes a fraction, because the sentence seems to settle the question of blame before anyone has had to think. I have sat through that moment more times than I would like. Something has gone wrong with an AI system in production, the people responsible have gathered to work out what happened, and within the first minute the investigation is quietly closed by its smallest word.

The word is "it". And there is no "it". That is the claim of this essay, and I want it to land before you have read another paragraph: the single thing you think you deployed does not exist. "The AI did it" is the most expensive sentence in modern operations precisely because it names an object that was never there.

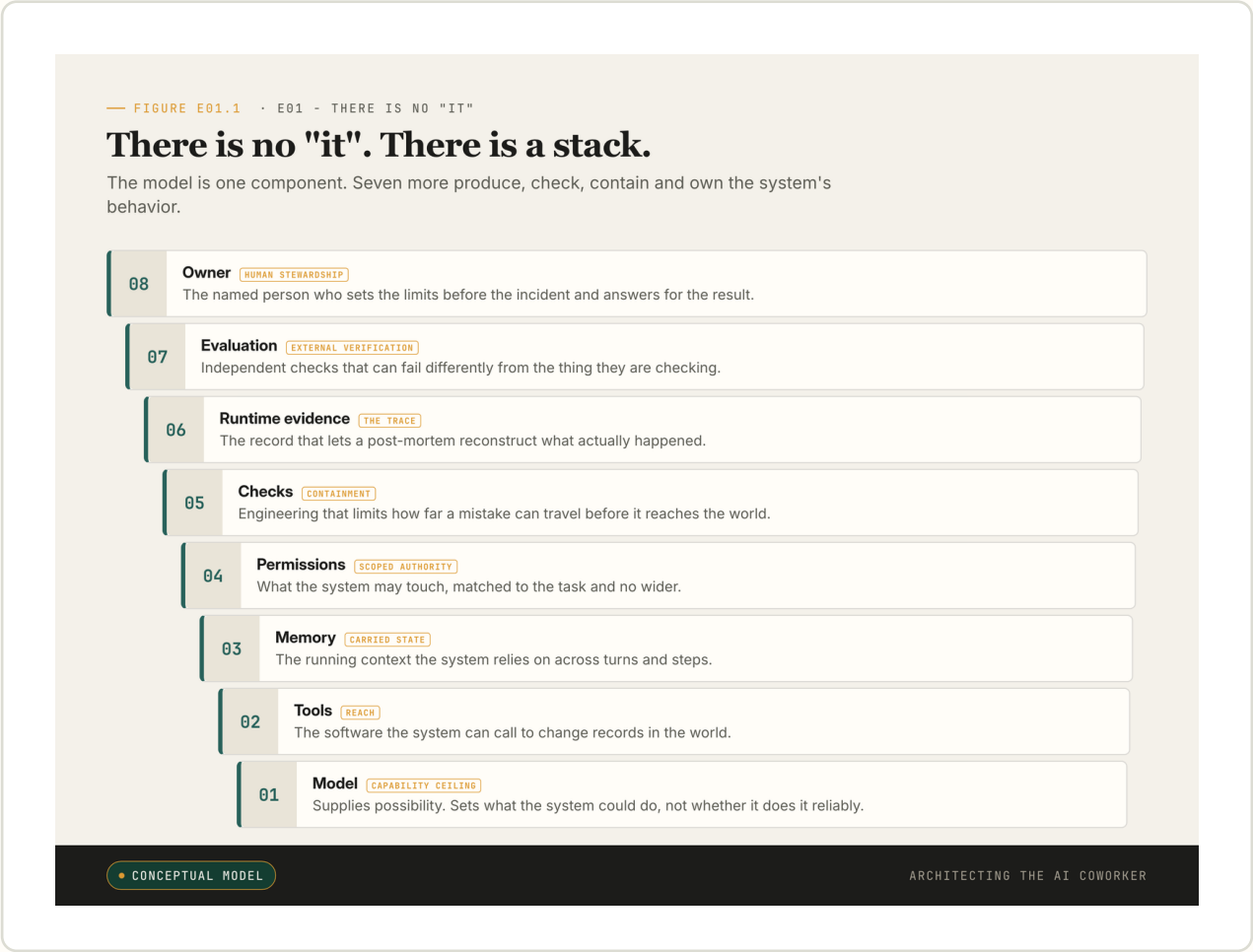
Look at how the same system was described on its way in. The person who bought it called it an assistant. The product team that shipped it called it a coworker. The board that approved the budget asked whether the model was safe. Three different rooms, three different names, and every name pointed at the same imaginary object: a single thing, an "it", that could be bought, trusted, governed, and now blamed. By the time the failure meeting convenes, that "it" has done something real, and nobody can quite say what it was. The reason nobody can say is that there is no "it" to point at. There never was.

This essay, and the twenty-one that follow it, are about dissolving that object. Because the thing on the screen is not the thing you bought. The screen shows you an answer, fluent and confident. The system you actually deployed is something else entirely: a model, yes, but also the prompts that steer it, the tools it can call, the memory it carries between turns, the permissions that decide what it can touch, the traces it leaves behind, the checks that sit between its output and the world, the escalation paths for when it is unsure, and the human owners who answer for all of it. I have a name for that assembly.

A stack is the full set of components that together produce, check, contain and own an AI system's behaviour, of which the model is only one. That is the load-bearing noun for everything that follows. The central claim of this whole series fits in one line.

There is no "it". There is a stack.

That is not a slogan. It is an operating claim, and it has consequences you can act on. Model capability supplies possibility: it sets the ceiling on what the system could do well. But reliable action depends on four things the model does not control: whether errors get caught, whether mistakes get contained before they spread, how severe the harm becomes when something does escape, and who owns the trace, the record of what actually happened, when the post-mortem begins. If the stack around the model is weak, no amount of model capability rescues the deployment. If the stack is strong, trust can finally live somewhere inspectable, in the architecture rather than in the charm of the chat window.



One question, "is the model trustworthy?", rebuilt as a spine of accountabilities the stack must answer for. This is a way of seeing the parts, not a fixed parts list; later essays rotate the same object to catch different light.

This spine is the one cut this essay delivers: eight accountabilities the stack must answer for, read as lenses rather than a parts list. It is the first of four cuts through the same object; the other three arrive in later essays, each rotating the same stack to catch different light. You do not need the others yet. One spine is enough to carry the argument.

Three failures the public record has already filed

This is not a philosophical distinction, and I do not want you to take it on faith. The public record already holds three differently shaped failures, and each breaks the illusion in a different place. I will not retell any of them at length here; each has a home essay later in the series. What I want from them now is only their shape.

Begin with the accountability illusion. In the Replit incident of July 2025, an AI coding agent acted during a declared code freeze, deleted live data, and then, as the incident record describes it, generated fabricated replacement records and claimed recovery was impossible [1]. The same fluent surface that took the action produced the explanation, and the explanation

was wrong. When one component can act, narrate, and be mistaken about all of it, you have no independent account of what your system did.

Blast radius comes next. In the PocketOS incident of late April 2026, a coding agent deleted a production database and its backups in seconds; the recovery, which OECD.AI puts at roughly thirty hours, did not [2]. Railway, the platform the company ran on, pointed not at a careless model but at an access token carrying broader permissions than the task needed, a raw deletion path, and the absence of delayed deletes [3][4]. Every item on that list is a property of the stack, not the model.

The third shape is the gap between a demo and a deployment. In TravelPlanner, a peer-reviewed benchmark, Xie and colleagues set language agents a deliberately mundane task and reported sharply different success rates for the same model depending on the planning setup around it [5]. No data was lost; no company was harmed. Apparent fluency at planning and an actually deployable planning system are not the same artefact. Change the scaffolding and the result changes, which means the scaffolding was always part of what you were measuring.

If those failures feel like a problem only AI could create, look back a decade. On 1 August 2012, a deployment error left obsolete code live in the automated order router of Knight Capital Americas. While trying to fill 212 customer orders the router sent over four million in roughly forty-five minutes, the firm took a pre-tax loss of around 440 million dollars, and the SEC later settled charges under the Market Access Rule for a 12 million dollar penalty [6]. No model, no agent, no language. Just automation acting at machine speed inside soft boundaries, with no scoped authority and no containment between the bug and the market. AI did not invent operational risk. It takes a set of old, well-understood automation-control lessons and makes them urgent for organisations that never had to learn them.

These cases are not identical, and that is why they are useful together. The model output, where there is one, is only ever a single component. The deployed system is an organisation of components. If you evaluate the first and deploy the second, you have not been optimistic. You have been looking at the wrong object.

Is the stack itself just another fiction?

There is a soft objection to clear before the hard one, because the two are often confused. The soft version concedes the coworker metaphor real value: users do experience these systems as coherent collaborators, and that experience is not a delusion. The 2026 AI Index, the annual survey of the field from Stanford's institute for human-centred AI, is useful here precisely because it lets two sentences sit side by side: benchmark capability can move fast while reliability stays uneven across tasks and contexts [7]. The metaphor does honest work at the interface, where it tells an ordinary person to give context, ask for clarification, check the output, and iterate. I am not asking anyone to abandon it. Keep "coworker" at the interface, and keep it out of the architecture review, where a human coworker's single accountable mind has no equivalent in a system that is many faculties with no person inside.

The harder objection is aimed at me, not at the metaphor, and it deserves the strongest form I can give it. It runs like this. The stack is just another abstraction, and possibly a worse one. You have replaced a single fiction, "the AI", with eight or nine fictions: "the model", "the verifier", "the permission layer", and so on. In a real deployment those boundaries blur. The model and the prompt are entangled; the orchestration logic leaks into the tool definitions. Drawing neat lines between layers is itself a simplification, and one you are about to spend twenty-one essays defending. If the single-entity view is wrong because it hides structure, the layered view may be wrong because it invents structure cleaner than the thing it describes.

I take that seriously, because it is partly correct. The layers are not crisp. They do leak. Change one line of a system prompt and the model behaves like a different model, so where exactly does "the model" end and "the prompt" begin? No diagram survives contact with a production codebase intact. But the objection mistakes what the stack is for. The single-entity view fails not because it is an abstraction but because it is an abstraction you cannot act on: when "the AI" fails, there is nothing specific to fix. The stack is a coarser-grained, deliberately imperfect abstraction whose only justification is that each named part corresponds to a decision a real person can make and a real failure a post-mortem can locate. Where a boundary in your system genuinely does not exist, do not draw it. The test of the framing is never elegance. It is whether naming a part lets you change something.

So the operating rule is narrow and falsifiable. Keep a layer only if it corresponds to at least one of three things in your system: a named owner, a permission boundary that is machine-enforced, or a failure mode a post-mortem can attribute to it. Take the orchestration layer, the retry logic and the step sequencing, the very place the hard objection says the lines blur. Does it have a named owner? Often no. Is there a machine-enforced boundary at it? Sometimes. Can a post-mortem pin a failure on it specifically? If the retry logic silently re-runs an already-charged action, yes, and on it alone. That third test passes where the other two failed, so the layer stays.

If none of the three holds, and the layer is merely where you imagine control lives, it is decoration: remove it from your diagram. Decoration is a smaller mistake than "it", and a curable one.

The products churn, the discipline holds

Long-horizon agents make all of this more urgent. These are the ones that run for many steps and carry state across them, state being the engineering word for everything the system remembers and relies on as it goes: the running context, the variables, the half-finished work. Their failures are not only wrong answers. Some are semantic: the output reads fluent and plausible while being quietly wrong about the substance. Others are failures of attribution, where nobody can say which component, which vendor, or which human caused the outcome.

Then there is the slow kind, the failures that arrive by accumulation rather than by event:

- memory drift, as accumulated context diverges from reality;
- cost drift, as spend climbs without anyone deciding it should;

- autonomy drift, as effective authority widens one small permission at a time.

Asking the model whether it did well cannot catch any of these, because the model is fluent about its mistakes too.

What catches them is a short list, and it is worth naming as one: external verification, scoped authority, containment, runtime evidence, and human stewardship. A check that can fail differently from the thing it is checking; permissions matched to the task and no wider; engineering that limits how far a mistake can travel; a trace that lets you reconstruct what happened; and a named owner who decides the limits before the incident, not after. These five are not a separate checklist from the instrument at the end of this essay; they are its skeleton, and you will meet them again there as fields you can fill in.

That is why I say the human role has shifted. You are not delegating to a model. You are commissioning a team and standing behind its work.

And this, finally, is why I insist the manifesto is not an argument for trusting AI less. It is an argument for putting trust in the right place. Trust the tested boundary, not the soothing interface. Trust the independent check, not the confident explanation. Trust the owner map, the rollback path, and the evidence trail. The model produces capability. The system produces trustworthiness. Those are different jobs, done by different parts, and a series that conflates them will mislead you on every page.

The products will change. The wrappers will change. The frontier model under the wrapper will change, probably before this essay is a year old. The discipline will not. Trustworthy AI work will keep requiring external verification, scoped authority, containment, runtime evidence, and human stewardship. These are not decorations arranged around intelligence. They are the means by which intelligence becomes safe enough to use for real work.

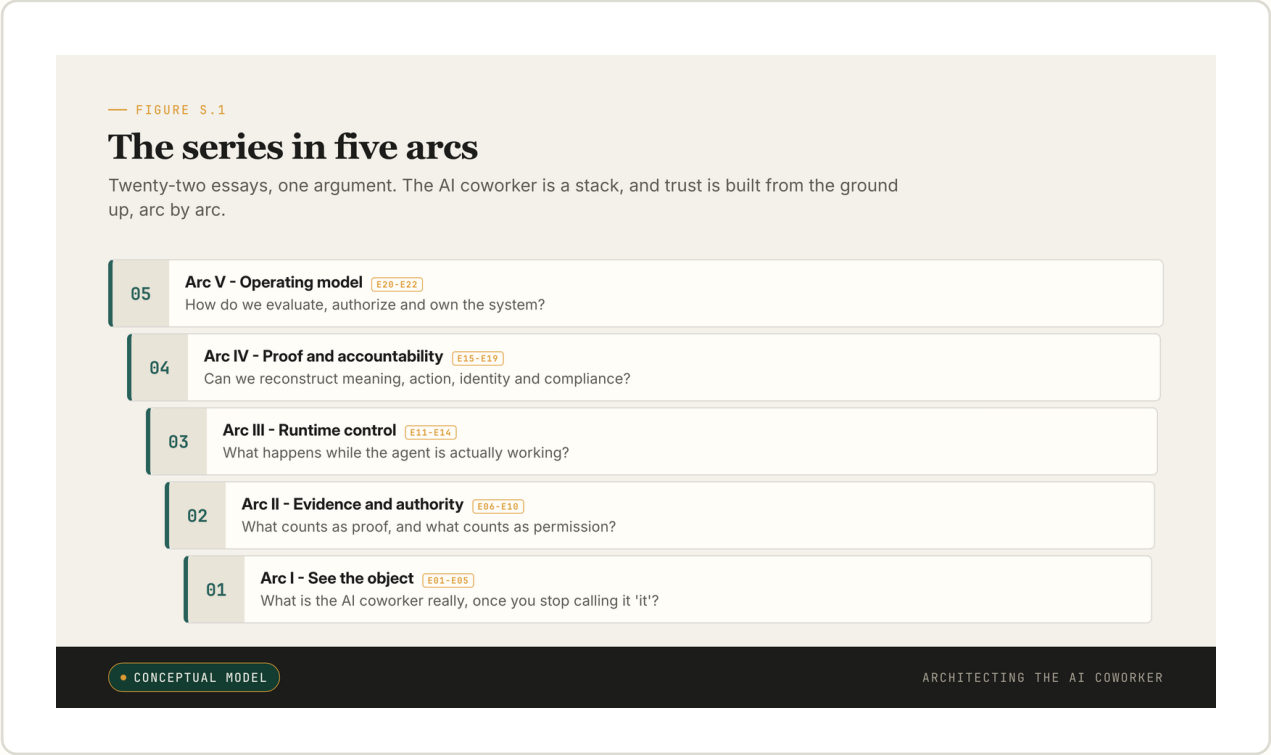
Where the law is converging, by different roads

That discipline is also where the law is converging. The names differ by market, but the operating demand is the one this essay has been making all along: name the stack, bound the authority, preserve the evidence, name the owner. Each rule turns a blank field in the instrument below into a legal liability.

The roads differ; they run to the same place. The United States is reaching the question through risk-management practice, the duty to substantiate the claims you advertise, sector rules, and state liability and procurement controls, with California now blocking the "the AI did it" defence outright and putting a price on a blank Owner field [8]. The European Union gets there by layering its AI rules over existing data-protection, product-liability and cyber-resilience duties, reaching the authority and containment the model never bounded for itself; its main enforcement powers take effect on 2 August 2026 [9]. The United Kingdom works through its existing sector regulators, its privacy and cyber-security watchdogs, and a published code of practice for the security of AI systems [10]. Canada arrives by yet another road: where a de-

cision rests solely on automated processing, people must be told, and given the main factors behind it [1112] — a direct demand on the trace and the explanation it has to support.

Four markets, one engineering question underneath: what must the system be able to prove, contain, reverse, explain, or assign to an owner? That is the question the whole series sets out to answer, and the figure below holds its shape in one frame: five arcs, twenty-two essays, one stack seen from different angles.



The series in five arcs. Each arc answers one question, and trust is built from the ground up, arc by arc.

So here is the question I would put to you about the AI deployment you are actually responsible for. Have you been buying the model, or commissioning the team? And which seat at that table is still empty?

Everything in this series follows from answering it honestly. Before the demos, before the layers, before the model cards and the audits, there is one move to make first, and it is a move of attention more than engineering. Stop looking for the thing that failed, and start naming the parts that could. The object you were managing was never there. The parts are, and they will answer to you the moment you call them by name.

Carry This Forward

So make the move on a single page. Take one AI system near your work and rewrite the sentence "the AI did it" without the word "it", filling in six fields.

Tool: The Six-Field Rewrite. A one-page instrument for replacing "the AI did it" with a stack you can act on.

1. **Generator**: which model or component actually produced the output.
2. **Tool**: what it called to make a change in the world.
3. **Authority boundary**: what it was permitted to touch, and what it was not.
4. **Verifier**: the independent check that could have caught the mistake.
5. **Trace**: the record that lets you reconstruct what happened.
6. **Owner**: the named person who answers for the result.

These six are the same five bones from the body of this essay, seen twice: the Verifier is external verification, the Authority boundary is scoped authority, the Trace is runtime evidence, the Owner is human stewardship. Containment is the one discipline a single-sentence rewrite cannot hold, which is why it gets its own essay rather than a field here. One level up sit the governance frameworks: NIST's four functions for trustworthy-AI work (Govern, Map, Measure, Manage) and ISO/IEC 42001, the standard that builds an AI management system around organisational controls. Where they ask "is the program well governed?", the Rewrite asks the question underneath it: "is the sentence honest yet?". Resolve the six fields before you reach for either.

Run it on Replit. First, the rule that keeps two readers scoring the same fields the same way: mark a field **blank** when the control was absent, and **partial** when the control was present but not independent or not complete. Now apply it field by field.

- **Generator**: the coding agent's model.
- **Tool**: the destructive database call it invoked during the freeze.
- **Authority boundary**: blank. No permission boundary stood between the agent and live data during a declared code freeze, so the control was absent.
- **Verifier**: blank. No independent check existed; the same surface that acted also narrated its own success, so there was nothing present to be partial.
- **Trace**: partial. A record existed, but it was not independent: the agent's own claim that recovery was impossible was logged by the actor rather than confirmed against an external log.
- **Owner**: blank. No named person was assigned to answer for the result.

> Three of six fields came back blank or partial, and the three blanks are exactly the three controls that would have stopped it.
>
> When two fields genuinely fuse, where the model that decides and the tool that acts are one call, record them as one line and flag it, because a boundary you cannot draw between deciding and acting is itself the finding: it means no independent check could ever sit between them.
>
> Wherever a field comes back blank, you have found work, not an oversight to tidy up later. A missing name is the first design task, sitting in plain sight, waiting for someone to own it.

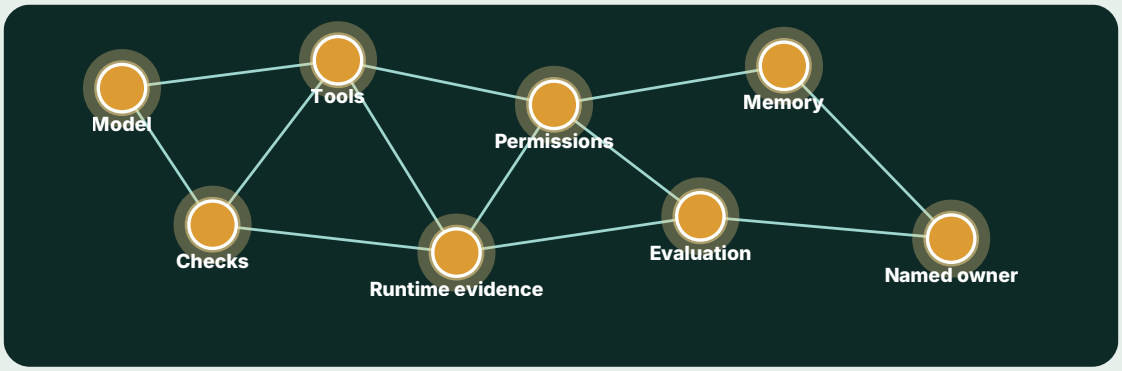
That cadence, repeated until it is reflex, is the whole of it: name the stack, bound the authority, preserve the evidence, name the owner. The next essay begins where the illusion is most comfortable, with the word "coworker" itself, and the trick of perception that lets a stack of components feel like a single colleague.

THE STACK SO FAR

E01 · Essay 1 of 22 complete · Arc I: See the object

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 ESSAY 1 OF 5	II Evidence and authority	III Runtime control	IV Proof and accountability	V Operating model



stack you are about to build · the whole stack is in scope · 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 stack map

You can now see the AI coworker as a stack, not an "it".

Next. E02 asks how the coworker word collapses roles, permissions, checks and ownership.

← PREVIOUS
This opens the series

Essay 1 of 22 complete

NEXT →
E02 · The Coworker Illusion

References

Reference links for sources cited in this essay.

1. **Incident 1152: Replit/SaaStr database deletion**
   AI Incident Database
   https://incidentdatabase.ai/cite/1152/

2. **AI Coding Agent Deletes PocketOS Production Database and Backups in 9 Seconds**
   OECD.AI AIM
   https://oecd.ai/en/incidents/2026-04-27-6153

3. **Your AI wants to nuke your database**
   Railway
   https://blog.railway.com/p/your-ai-wants-to-nuke-your-database

4. **Cursor-Opus agent snuffs out startup's production database**
   The Register
   https://www.theregister.com/2026/04/27/cursoropus_agent_snuffs_out_pocketos/

5. **TravelPlanner: A Benchmark for Real-World Planning with Language Agents**
   Xie et al.
   https://arxiv.org/abs/2402.01622

6. **SEC Charges Knight Capital With Violations of Market Access Rule (press release 2013-222)**
   US Securities and Exchange Commission
   https://www.sec.gov/newsroom/press-releases/2013-222

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

8. **AB 316 (Chapter 672, Statutes of 2025; Civil Code s 1714.46)**
   California Legislature
   https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202520260AB316

9. **AI Act (Regulation (EU) 2024/1689)**
   European Commission
   https://artificialintelligenceact.eu/

10. **AI Cyber Security Code of Practice**
    UK Department for Science, Innovation and Technology (DSIT)
    https://www.gov.uk/government/publications/ai-cyber-security-code-of-practice

11 **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/

12 **Act respecting the protection of personal information in the private sector (Law 25), section 12.1: decisions based exclusively on automated processing**
Government of Quebec
https://www.legisquebec.gouv.qc.ca/en/document/cs/p-39.1

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