03

THE STACK BEHIND THE AI COWORKER

# Demo Is Not Deployment

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

A polished demo wiped $100bn off Alphabet before anyone asked if it could ship. Six gates ask first.

An essay in the series **Architecting the AI Coworker**.
Approx. 19 minute read · Essay 03 of 22

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

> On 8 February 2023, Reuters reported that Google's new chatbot, Bard, had given an inaccurate answer in a promotional video. Asked about the James Webb Space Telescope, it claimed the telescope took the first image of a planet outside our solar system, which is wrong [1]. The clip had been built to impress. It did the opposite. By the close of trading, Alphabet had lost roughly $100bn in market value [1]. No customer had been harmed. No system had been deployed. A polished demonstration, watched by enough people, had simply overpersuaded, and then under-delivered, before any deployment authority existed at all.

That is the lesson in its sharpest form, and it does not depend on the size of the company or the year on the calendar: a demo can move a room, or a market, long before anyone has decided whether the underlying system can be trusted to act. Hold that fixed point. The numbers later in this essay will illustrate it; they will not replace it.

Now picture the smaller version, the one you have almost certainly sat through. A team is at the front of the room. They run an agent live: it takes a messy brief and produces, in about ninety seconds, a clean multi-step plan with costs attached. Someone senior leans forward. Someone else stops checking their phone. And before the run has even finished, a voice from the table asks "so when can we turn it on", and the meeting has quietly changed, without anyone deciding it, from an evaluation into a launch plan. The last essay broke the word "it" into a stack of named roles and gave you a test to run before you trust an AI system with real authority. The illusion does its most persuasive work earlier than any procurement meeting. It does it in exactly that moment: the demo runs cleanly, the room nods, and a question gets skipped.

The skipped question is the subject of this essay, so let me name the assumption underneath the nod and then take it apart. The agent produces a beautiful plan, or a working feature, or a polished brief, in front of an audience, and the audience reads that clean run as evidence the system can be deployed. It is not. A demo proves that a capability can appear under arranged conditions. Deployment has to prove that the same capability survives real users, stale state, permissions, partial failures, costs, audit requirements, stopping rules, and consequences. The mistake is treating the first as evidence for the second, and almost everyone in that room made it without noticing.

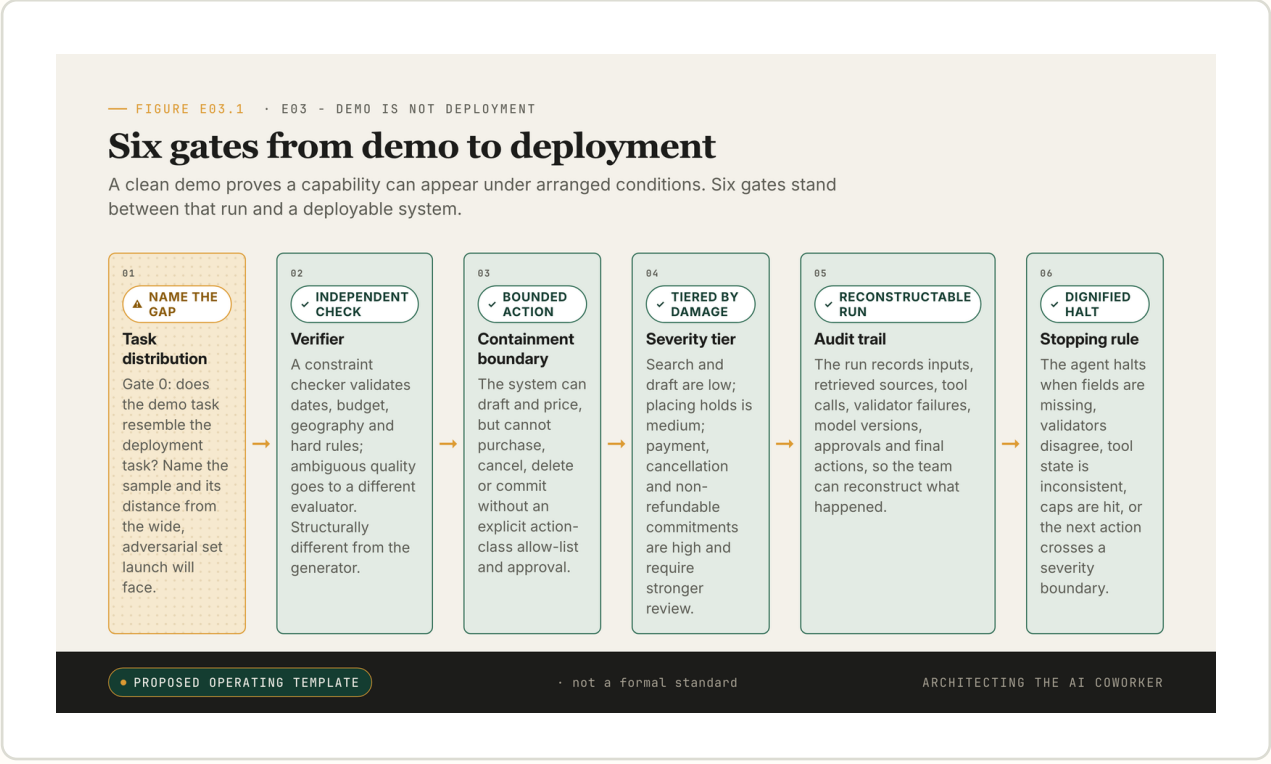
I want to anchor the timeless claim with two numbers, because numbers make a gap concrete. Read them as illustration, not as the argument itself.

*TravelPlanner, 2024 paper, GPT-4 Turbo:* zero point six percent in the two-stage tool-use setting, and four point four percent when the required information is handed to it up front.

Both come from TravelPlanner, a benchmark, meaning a standard test used to compare systems, built by Xie and colleagues around a deliberately mundane task: assemble a coherent

multi-day trip under real constraints [2]. The setting around a model, comprising the tools it is given, the information it is handed, and the checks wrapped around it, is what engineers call the harness. The smaller number is GPT-4 Turbo's final-pass rate in the two-stage tool-use setting; the larger is the same model's pass rate when the required information is handed to it up front. Both are dated measurements from the 2024 paper, not a permanent verdict. Hold them lightly until the six gates below, and especially Gate 0 (representative task distribution), tell you what the numbers were measuring in the first place.

Read those two figures slowly, because they are not arranged the way intuition expects. The more elaborate setting, the one with the tool-use loop, scored lower than the stripped-down one. A harness is not a magic additive; it is a set of design choices, and a two-stage tool-use loop can introduce its own failure modes (a retrieval step that returns the wrong thing, a reasoning step that compounds an early error) that a simplified setting never gives the model the chance to make. The honest reading is the lesson with two data points underneath it: a fluent plan and a deployable planning system are different objects, and the harness you wrap around a model decides which of the two you are actually measuring.




*A clean demo is useful evidence, and incomplete. Gate 0 asks whether the demo task resembles the deployment task; Gates 1 through 5 stand between it and a deployable system.*

## Change the harness, change what the number measures

TravelPlanner is useful precisely because the task feels almost aggressively mundane. A travel plan is not grand strategy. It is dates, cities, budgets, transport, opening hours, meals, preferences, and constraints that collide the instant one thing changes. That is exactly what makes it

a good x-ray. A model can sound like a competent travel agent while quietly producing an itinerary that cannot exist: the prose is fluent, and the plan is impossible [2].

The result needs to be stamped carefully, and I want to do that honestly rather than wave it through. It is a measurement from a particular benchmark on particular dates, not a law of nature, and not a claim about every system built since. What it shows is how sharply the harness matters: two-stage tool use, simplified direct planning, supplied information, and validators can each change what is being measured. That is not a weakness in the argument. It is the argument. Engineering work on long-running agents starts from exactly this fact, building distinct planner, generator, and evaluator roles around the model [2][3]. When the harness changes the result, the deployable unit was never the naked model. It was the model plus the harness.

The slide says "model". The run depends on retrieval, routing, tool wrappers, validators, prompts, sandboxes, approvals, retries, audit logs, and someone who knows where the awkward edges are in the test environment. A serious demo may contain all of those. A theatrical demo may contain only the ones that keep the recording clean. From the audience side, the two can look identical, which is the whole problem.

That is why "can it do the task?" is the wrong first question. Of course it can, sometimes. The better question is narrower and far more revealing: under what system conditions did it do the task, and which of those conditions will still exist after launch?

## What a demo is built to hide

WebArena and OSWorld report the same pattern: realistic, long-horizon, stateful tasks reveal large gaps between agent and human success [4][5]. Deployment changes the object being evaluated. The range of tasks widens. Users give partial instructions. Tools return inconsistent state. APIs time out. Permissions leak across environments. The system has to decide, in the moment, whether to stop, escalate, retry, spend more, or act. A model-card benchmark score does not answer those questions, and usually was never designed to.

The durable part is the shape, and it does not move with the model of the month. The numbers move; the shape does not, because every capability gain widens the set of tasks the system is trusted with. So the gap the gates name is between arranged and adversarial conditions, not between weak and strong models; a better model just meets the gap on harder tasks. When an environment becomes interactive and stateful, so each action changes the world and the next step has to cope with what the last one left, the distance between an impressive output and finished work becomes visible.

A demo is a constructed artefact, and construction is the entire point. The dataset is chosen, the route through the product is known, the integrations are warmed up, and failure has low consequence. A good demo is scoped evidence. A bad procurement process treats scoped evidence as deployment evidence, and pays for the difference later. A demo is often built, deliberately or not, to hide exactly the dimensions those benchmarks expose.

The industry's response to this has not been to abandon models. It has been to build harnesses. Anthropic's engineering writing on long-running application-development agents describes planner, generator, and evaluator roles, structured handoff artefacts, and harness design around the model [3]. Enterprise-evaluation work such as CLEAR similarly treats long-horizon agents as systems whose cost, latency, efficacy, assurance, and reliability have to be evaluated together, rather than collapsed into one accuracy number [6]. Again: not model versus system. Model inside system.

## The Six Gates

> **Tool: The Six Gates.** A practical instrument for stopping a room from confusing a clean run with a deployable system. Each gate names a property a demo can hide and a deployment must answer for. Walk them in order.
>
> 1. **Representative task distribution**: does the demo task resemble the deployment task?
> 2. **Verifier**: is there a check structurally different from the generator?
> 3. **Containment boundary**: what can the system do without explicit approval?
> 4. **Severity tier**: are actions graded by how much damage they can do?
> 5. **Audit trail**: can the run be reconstructed after a dispute?
> 6. **Stopping rule**: does the agent know how to halt rather than improvise?

The Six Gates are not a certification scheme, and they do not duplicate the security and governance frameworks teams already use. A vulnerability catalogue (the OWASP Top 10 for LLM Applications is the familiar one) asks which attacks could land. A lifecycle governance framework (NIST AI RMF and its kin) asks whether the wider program is well run. The Six Gates work at a single decision point upstream of both: the moment a room decides whether a demo is evidence of deployability. They ask the narrower question both of those presume an answer to: is the task you just watched a sample of the task launch will face, and do the five system properties exist that turn a clean run into a survivable one?

There is a gate that comes before all the others, and almost no demo room asks about it. Call it Gate 0: representative task distribution. Before you ask whether a system verifies, contains, or stops, ask whether the task you just watched resembles the task you intend to deploy. The Bard clip failed here first. The James Webb question was a single, hand-picked prompt; the deployment surface was every question a search user could type. A demo task is, by construction, drawn from a narrow and favourable distribution. The deployment task is drawn from a wide and adversarial one. If the demo task and the deployment task are different distributions, every later gate is measuring the wrong thing, however cleanly it passes. So before verification: name the task distribution the demo sampled from, and how far it sits from the one launch will face.

"How far" sounds soft, so make it ratable on four concrete dimensions:

- *Input source*: was the input curated and hand-cleaned, or live user text with its typos, omissions, and contradictions?
- *State freshness*: was the environment warmed and pre-loaded, or stale the way production state is when a cache expired an hour ago?
- *Failure injection*: did the run face any tool timeouts, permission denials, or malformed responses, or was every dependency healthy by arrangement?
- *Prompt selection*: was the prompt hand-picked, or drawn at random from a real query log?

A demo that is favourable on all four (curated input, warm state, no injected failure, hand-picked prompt) is sampling the easiest possible sliver of the surface, and sits maximally far from the distribution launch will face. Each dimension a demo concedes is one notch closer to deployment, and the gap is something you can now point at rather than assert.

The remaining five gates apply once Gate 0 holds, and the table below sets out all six in order. Use them on a TravelPlanner-style task: "Plan a five-day trip for a family of four, under a budget, with dietary constraints, one museum closed on Wednesday, no late-night arrivals, and a refundable booking preference." This is a reference example, not a reproduced TravelPlanner instance. TravelPlanner supports the general claim that multi-constraint trip planning is a realistic planning benchmark; the table below is an operational artefact built from that benchmark's shape, not data from any real run [2]. One term in it is worth fixing in advance: an action-class is a category of action grouped by how much damage it can do, with searching and drafting in one class, placing a hold in another, and paying or cancelling in a third, so that permissions and review can be set per class rather than per individual click.

| PRODUCTION-GRADE QUESTION | DEMO ANSWER THAT IS NOT ENOUGH | PRODUCTION-GRADE ANSWER |
|---|---|---|
| 0. What is the task distribution? | "Watch it plan this trip." | The demo used one pre-vetted city pair with cached availability and a complete brief; deployment faces arbitrary origin and destination, live inventory that changes between the search and the hold, and users who omit the budget or the dietary constraint entirely. The demo sampled the easiest sliver of the surface, and that distance is stated, not assumed. |

| PRODUCTION-GRADE QUESTION | DEMO ANSWER THAT IS NOT ENOUGH | PRODUCTION-GRADE ANSWER |
|---|---|---|
| 1. What is the verifier? | "The model checks its own answer." | A constraint checker validates dates, budget, geography, opening hours, and hard preferences; ambiguous quality is sampled by a different evaluator or a human reviewer. The verifier is structurally different from the generator. [36] |
| 2. What is the containment boundary? | "It only books when the user asks." | The system can draft and price an itinerary automatically, but cannot purchase, cancel, delete, refund, or commit without an explicit action-class allowlist and approval boundary. |
| 3. What is the severity tier? | "All travel actions go through the same flow." | Search and draft are low-severity; placing holds is medium-severity; payment, cancellation, passport data, and non-refundable commitments are high-severity and require stronger review. |
| 4. What is the audit trail? | "We can see the final itinerary." | The run records inputs, retrieved sources, tool calls, validator failures, model versions, approvals, and final actions, so the team can reconstruct what happened after a dispute. [3] |
| 5. What is the stopping rule? | "The agent keeps working until it is confident." | The agent halts when required fields are missing, validators disagree, tool state is inconsistent, budget or latency caps are hit, or the next action crosses a severity boundary. |

This table is deliberately boring. Boring is where deployment lives.

The table also gives the instrument an output. Score each gate present, partial, or absent. If Gate 0 is absent, stop: the rest of the gates are measuring the wrong task, and a clean pass on them is worse than no pass at all, because it manufactures false confidence. If any of Gates 1 through 5 is absent on a high-severity action class, whether the payment, the cancellation, or the irreversible commit, what you have is a case for exploration funding, not a launch date. Only when all six are present and disclosed does the launch become something you can put a

date on. The instrument does not score a model; it scores whether the room is allowed to stop talking about accuracy and start talking about a date.

And this is exactly what the synthetic table and the real 0.6 percent number have to say to each other: the table is the harness whose absence that number measures, because nothing in a model-only run rejected the impossible itinerary before it became the answer. That is why the figure being "not data from any real run" is a strength, not a hedge. The table is the missing machinery, named in full; the 0.6 percent is what the world looks like when it is not there.

Watch the difference it makes. A model-only version of the trip task may produce a fluent itinerary with a flight that lands after the rental desk closes, a restaurant that cannot handle the allergy, or a hotel that breaks the budget once taxes are added. The final answer looks like finished work. The validators would fail it. A production-harnessed version changes the run entirely. The planner decomposes the task. Retrieval brings in current options. Tool calls price the plan. Deterministic checks, fixed rules that give the same answer every time, reject hard constraint violations. A separate evaluator or human reviewer handles the genuine judgement calls. The permission boundary keeps the system in draft mode until the user approves a high-severity action. The audit trail records the path. The stopping rule prevents the system from quietly improvising around facts it does not have. The visible answer may be almost the same length. The deployed system is a different machine.

## Production-grade means errors have somewhere to go

Production-grade does not mean "the model is accurate". It means errors have somewhere to go before they reach the world. Read the gates as a chain, because that is what they are. Gate 0 makes sure the chain is even pointed at the right task. After that the verifier catches the error that would otherwise survive unnoticed; if one slips past it, the containment boundary keeps that unnoticed error from acting; whatever still escapes runs into the severity tier, which caps the damage it can do, while the audit trail keeps the whole run legible after the fact; and when none of that is enough, the stopping rule hands the system a dignified way to fail instead of a confident way to compound the mistake. Read down that list and notice what kind of thing each item is: not one of them is a property of the model, and not one of them appears in the accuracy benchmark you are being shown, because they are all properties of the system wrapped around it.

That is also why demo evidence so reliably overpersuades leadership teams, and the mechanism is worth stating exactly: a demo's persuasive power scales with the production fragility it conceals. The more carefully a happy path is staged, the more failure surfaces it had to hide to stay clean, so polish and hidden risk move together, and the smoothest run in the room is often the one with the most still unaccounted for. The demo shows the happy path and the model output; what it hides is exactly that chain. But those unglamorous pieces are what makes a system survivable. If a demo cannot answer the five questions, it may still be worth funding as exploration. It should simply not be treated as production proof.

## What the demo can and cannot buy

An honest reader will push back on the claim that demos prove little, and the push-back is sound. Demos and benchmarks are not fake, and they are not worthless. Without them, teams would have no scalable way to compare systems early. A live demonstration can expose interface problems quickly; a benchmark can force measurable progress; and a model that cannot succeed even in a clean setting is unlikely to succeed in a messy one. The strongest version is sharper still: a polished demo cheaply screens out incompetent vendors, because a team that cannot stage a clean run with a known dataset on a chosen route is almost certainly unable to stage a deployment either. That is correct, and useful. A demo that fails on its own home ground is a strong negative signal.

So the procurement mistake the Six Gates correct is not treating the demo as signal at all. It is treating the absence of failure under arranged conditions as evidence of presence under adversarial ones. Those are different epistemic moves, and only the first is what a demo can support. A vendor that fumbles the demo has told you something real; a vendor that passes it has only earned the right to be asked the six gate questions, not yet the right to be deployed.

The answer, then, is not to sneer at demos but to assign them the right evidentiary weight, and to say the conversion line out loud: demos are capability evidence; only the six gates convert that into deployment evidence. A demo is scoped capability evidence. A benchmark is comparative evidence under a defined protocol. Neither is deployment evidence unless the protocol includes the deployment properties that actually matter: repeated runs, realistic state, tool failures, permissions, verification, containment, human review, cost, latency, and auditability. CLEAR's whole premise is that enterprise agent evaluation needs multiple dimensions, not a single headline score [6]. The danger was never the demo. The danger is the unmarked conversion rate between "we saw it work" and "we have engineered what happens when it fails".

And the strongest objection, met head-on: am I demanding a finished deployment just to evaluate one? No. The gates are questions, not build orders, and Gate 0 along with the five that follow can be answered on paper before a single line of harness is written. What they forbid is funding a launch as though the answers already exist when no one has been asked to give them. The cost they add is the cost of naming your failure surfaces out loud, which is cheap; the cost they prevent is the $100bn kind, which is not.

Regulators come down on the same side, for a reason worth stating plainly: in the eyes of the law, the company that ships the system is the one making the claim, and an AI demo earns no gentler standard than any other piece of marketing. In the UK, the advertising rules are media-neutral, so the fact that a machine generated the claim does not move responsibility off the advertiser [7]. Across the EU, under the unfair-commercial-practices rules, a trader's message is judged by its effect on the average consumer, whatever component produced it. In the US, the consumer-protection regulator's enforcement push under Section 5 requires AI capability claims to be backed by evidence at the moment they are made, and has treated demo-derived performance claims as deceptive where the underlying system cannot meet them in produc-

tion [9]. And in Quebec, the privacy regulator expects any workplace AI deployment to rest on a written AI policy, an impact analysis of the system, and genuine involvement of the affected employees [8]. One rule under four flags: the clean run is the claim, not the proof.

Before the callback, one honest stock-take, with the settled part set against the uncertain. TravelPlanner gives clean, setting-sensitive data points, dated to its 2024 paper: GPT-4 Turbo at 0.6 percent final pass in the two-stage tool-use setting and 4.4 percent in the simplified direct setting with required information supplied [2]. WebArena and OSWorld show the same broad pattern in web and desktop environments, where realistic, long-horizon, stateful tasks revealed large gaps between agent and human success in their original benchmark reports [45]. And the evaluation and engineering literature now treats harnesses, evaluators, cost, latency, assurance, and reliability as part of the system under evaluation, not as decoration around the model [36].

Against that, three things stay genuinely unknown: whether today's private frontier systems would reproduce those old benchmark numbers under the exact same scaffold, a moving target nobody outside the labs can settle; whether any particular vendor's demo carries a serious verifier, containment boundary, or stopping rule, which cannot be known at all unless those controls are disclosed or independently inspected; and the true production incident rate, which stays hidden because so many failures are private, quietly fixed, or legally constrained. So the honest posture is restraint, which is exactly the question to carry back into the room.

So go back to that demo room, the one where the voice at the table asked "so when can we turn it on". The honest answer is not yes and not no. It is a question asked back, before anyone discusses accuracy at all: tell me how close this demo task is to the task launch will face, and then name the verifier, the containment boundary, the severity tier, the audit trail, and the stopping rule for the exact action we just watched. If the answers are crisp, the launch can become a date. If the answer is "the model is getting better", you have just learned what the demo was hiding, and the launch is still a piece of engineering nobody has done yet. Bard's promotional clip never reached that question. The $100bn was the price of skipping it.

## Carry This Forward

**One move, before the next demo.** Take the last demo that impressed you and answer Gate 0 in a single sentence: name the task distribution it sampled from, and how far that sits from the task launch will face. Then ask the five questions for the highest-severity action you watched, the verifier, the containment boundary, the severity tier, the audit trail, the stopping rule. Crisp answers buy a date. Missing answers are not a procurement footnote; they are design work with names, and a system cannot be made reliable until those failure surfaces are named. A demo is scoped capability evidence and nothing more; only the gates convert it into deployment evidence.

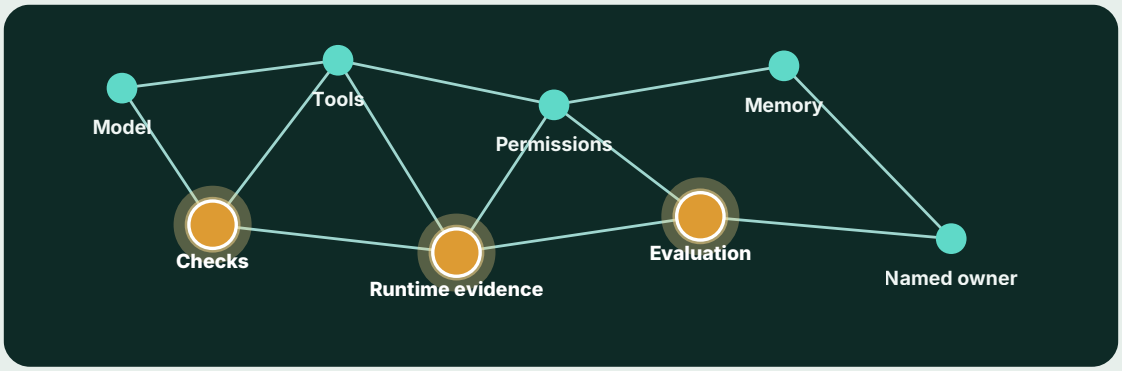
*Next:* the hidden architecture those gates kept pointing at, foundation, alignment, reasoning, tools, routing, planning and memory, stopping economics, orchestration, and governance, laid out on a single page, layer by layer, so you can see where every failure mode actually lives. Demo is not deployment because deployment is the stack.

## THE STACK SO FAR

E03 · Essay 3 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<br>ESSAY 3 OF 5 | II Evidence and authority | 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.**

### Demo-to-deployment gates

You can now separate demo evidence from deployment evidence.

**Next.** E04 asks where a deployed agent actually breaks.

← PREVIOUS
E02 · The Coworker Illusion

Essay 3 of 22 complete

NEXT →
E04 · The Nine Layers Where Agents Break

# References

Reference links for sources cited in this essay.


1. **Alphabet shares dive after Google AI chatbot Bard flubs answer in ad**
   Reuters
   https://www.dawn.com/news/1736179

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

3. **Harness design for long-running application development**
   Anthropic
   https://www.anthropic.com/engineering/harness-design-long-running-apps

4. **WebArena: A Realistic Web Environment for Building Autonomous Agents**
   Zhou et al.
   https://arxiv.org/abs/2307.13854

5. **OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments**
   Xie et al.
   https://arxiv.org/abs/2404.07972

6. **Beyond Accuracy: A Multi-Dimensional Framework for Evaluating Enterprise Agentic AI Systems (CLEAR)**
   Mehta, Sushant
   https://arxiv.org/abs/2511.14136

7. **Generative AI & Advertising: Decoding AI Regulation**
   UK Advertising Standards Authority / CAP
   https://www.asa.org.uk/news/generative-ai-advertising-decoding-ai-regulation.html

8. **Brief to the Quebec Ministere du Travail on AI in the workplace**
   Quebec Commission d'acces a l'information
   https://www.cai.gouv.qc.ca/

9. **Operation AI Comply (Section 5 enforcement against AI capability claims)**
   US Federal Trade Commission
   https://www.ftc.gov/news-events/news/press-releases/2024/09/ftc-announces-crackdown-deceptive-ai-claims-schemes

# 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 |