10

WHITE PAPER · AI OPERATING GOVERNANCE

# Confident and Wrong

Challenge protocols for plausible AI output

**Marco Policani**

Enterprise Portfolio · PMO · AI Operating Governance

policani.net · linkedin.com/in/marcpolicani

July 2026

EXECUTIVE SUMMARY

**Plausible output needs a challenge protocol before it changes a consequential decision or action.**

The hardest AI error to catch is coherent, specific, aligned with the reader's expectations, and wrong in a way that survives a quick read — and agreement routes the least scrutiny to exactly those answers. This paper converts 'have a human review it' into a five-part challenge protocol with declared decisions, real disconfirmation, independent checks, accountable reviewers, and failures that teach.

01 Sort output by declared consequence before review, so scrutiny concentrates where reliance does.

02 Follow load-bearing claims to ground outside the answer; asking the model whether it is sure is corroboration theater.

03 Watch the rate at which independent checking changes claims — the workflow's real error rate, and almost always above the prior.

## The argument

Fluent AI output can enter consequential work with more confidence than evidence. A challenge protocol turns vague advice to verify into a repeatable control that states the decision, seeks disconfirmation, checks claims independently, records review, and learns from failure.

The generated analysis that came closest to reaching one of my steering decks was excellent in every visible respect. Clean structure, plausible figures, a benchmarking comparison that flattered our position, and a citation to an industry report I recognized. One number felt slightly convenient, so I opened the report. The report existed; the number did not — the model had blended two adjacent statistics into one that said what the narrative needed. Nothing about the document signaled the fabrication. The signal was only that it agreed with us a little too well, and that I had, that week, the time to be suspicious.

That is the governance problem in miniature. The hardest AI error to catch is not nonsense — nonsense filters itself. It is the answer that is coherent, specific, aligned with the reader's expectations, and wrong in a way that survives a quick read. Plausibility lowers the friction of reliance, and agreement lowers it further: an output that confirms a preferred interpretation gets less scrutiny than one that creates resistance, which means error-checking effort is allocated in exactly the wrong direction — most scrutiny for the answers that annoy us, least for the answers that flatter us.

'Have a human review it' is the stock answer, and as stated it is not a control; it is a hope with a name attached. Review that works has structure: it knows what decision the output feeds, it hunts for what would disprove the answer rather than polishing it, it follows claims to sources, it is performed by someone with the time and standing to say no, and it leaves a trace that teaches the organization something.

The agreement dynamic is the quiet engine underneath. Confirmation is pleasant, disconfirmation is work, and generated output arrives pre-shaped to the asker's framing. The bias has no villain — users are not credulous, models are not malicious — and yet the pipeline reliably routes the least scrutiny to the most agreeable claims. Controls must be designed against that gradient, because individual conscientiousness does not hold against it at volume.

This paper — for AI program owners, risk leaders, workflow owners, and executives who rely on generated analysis — turns that structure into a five-part challenge protocol. OpenAI's public account of a sycophancy incident and Microsoft's lessons from red-teaming a hundred generative products both show model behavior failing in interaction, not just in the abstract [1][2]; the protocol is what my own reviews hardened into after enough convenient numbers.

## Why plausible error beats casual review

Casual review is calibrated to human failure patterns, and generated output fails differently. When a colleague's analysis is weak, the weakness usually shows at the surface: gaps in logic, hedged language, formatting that betrays hurry. Fluent generation inverts every one of those signals. The prose is confident, the structure is textbook, the hedges are absent — and none of that correlates with correctness. A reviewer's lifetime of calibration, in which polish predicted care, becomes actively misleading.

The failure compounds through iteration. A user who receives a nearly-right answer refines it: asks for elaboration, requests a stronger version, feeds it back for polish. Each round makes the artifact more persuasive without touching the underlying premise. By the third pass, the original unsupported claim is load-bearing scaffolding beneath paragraphs of derived content, and the person most familiar with the document is the person least able to see it — they built their understanding on the same premise the document did.

Volume finishes the job. Generation is cheap, so output multiplies; review capacity is human, so it does not. Without a protocol that concentrates scrutiny where consequence lives, the ratio of checking to producing degrades quarter by quarter, invisibly, until an error with real consequence finally surfaces — at which point organizations typically respond by mandating review of everything, which collapses under its own load within months and returns the system to hope.

## The five-part challenge protocol

The protocol is five questions asked in sequence, proportionate to consequence. It is deliberately not a checklist for every generated sentence — low-stakes assistance should stay frictionless. It is the standard path for output that could change a consequential decision or action, and each part exists because a specific, observed failure defeats review without it.

EXHIBIT 1

### Five challenges stand between fluent output and consequence

1 **State the consequential decision**

Review intensity should follow what the output may change, not how polished the text appears.

2. **Seek disconfirming evidence**
   The first challenge should test the favored conclusion rather than ask the system to make it sound stronger.
3. **Check claims independently**
   Verification means following important claims to sources outside the generated answer.
4. **Use a named reviewer and escalation path**
   Human review works only when the reviewer has time, competence, authority, and a defined response to uncertainty.
5. **Learn from plausible failures**
   A rejected answer is operating evidence about prompts, data, interfaces, review load, and permitted use.

## State the consequential decision

Challenge begins by naming what the output might change: which decision, affecting whom, how reversible, at what cost if wrong. Until that is stated, review intensity is set by the wrong variables — document length, deadline pressure, how polished the text looks — rather than by what is actually at stake.

The failure this prevents is reviewing an artifact without knowing it is an input to something that matters. A reviewer copy-edits a generated summary as prose; three weeks later its headline figure anchors an investment recommendation. Nobody decided that figure deserved investment-grade verification, because at review time nobody knew — or recorded — where the summary was headed. Consequence was assigned by drift.

The operating practice is a use declaration above the output before review begins: decision affected, affected party, reversibility, error cost, and the mode of use — brainstorming, drafting, recommendation, or basis for action. The declaration takes one minute, sorts the output into the right review tier, and creates the record that makes the rest of the protocol auditable. Low-consequence modes exit here, by design; the protocol's first decision is whether the protocol applies.

Tiering makes the declaration concrete. A workable default: tier one, ideation and internal drafting — no protocol, full speed. Tier two, content informing a recommendation — disconfirmation pass plus spot verification of load-bearing claims. Tier three, content that will justify an action affecting money, customers, staff, or the public record — full protocol, named reviewer, verification log. The tiers are not sacred; what is sacred is that the sorting happens before review, on declared consequence, rather than after an incident, on regret.

## Seek disconfirming evidence

The natural instinct with a promising answer is to make it stronger — ask for more support, more polish, more confidence. The protocol's second move runs opposite: try to break it. What observation would make this conclusion false? What alternative explanation fits the same facts? Which stakeholder would dispute this, and what would their best argument be? Where does the answer hold only inside one framing?

The failure this prevents is the echo loop, and the sycophancy pattern makes it mechanical: models tuned toward agreeableness amplify the user's framing, so a user who prompts from a preferred conclusion receives ever-better dressed versions of that conclusion [1]. Each refinement feels like progress; the premise never gets touched.

Teams have run a dozen iterations deep in this loop, accumulating what looks like a well-developed analysis and is actually the same untested guess, twelve drafts more confident.

Practical disconfirmation is cheap and often uses the same tool: ask the model for the strongest case against its own answer, for the conditions under which the answer fails, for the data that would distinguish it from the leading alternative. Then check whether any of that material was already available and simply unexamined. A conclusion that survives a serious attempt to break it has earned something; a conclusion that has only survived attempts to improve it has earned nothing yet — and the reviewer should be able to say which kind is on the table.

A worked example of the move's cheapness: a generated competitive analysis concludes that a rival's pricing change signals retreat from the mid-market. The disconfirmation pass asks three questions in ten minutes. What would retreat predict that we could observe — hiring freezes, partner churn, feature deprecation — and do we observe it? What else explains the pricing change — a bundling strategy, a fiscal-year discount cycle? Who inside our own sales team has evidence against the retreat reading? In the case I am describing, the third question ended the thesis: two account executives had lost mid-market deals to the rival that month. The analysis had been one prompt away from a strategy meeting.

## Check claims independently

Verification means following important claims to ground outside the generated answer: opening the cited report, re-running the calculation, reading the actual policy clause, confirming the quotation. Inside the answer, everything is testimony from the same witness; checking the answer against itself — asking the model whether it is sure, requesting sources from the same session — is corroboration theater.

The failure this prevents is exactly my convenient-number story, and its family: citations to real sources that say something adjacent, quotations lightly paraphrased into new meaning, calculations correct in method and wrong in one input, policy interpretations that were true two revisions ago. Each artifact looks like evidence. Each survives a reviewer who checks that sources are cited without checking what the sources say.

Proportionality keeps this affordable: not every claim, the load-bearing ones — selected by consequence, the ones on which the decision actually pivots. For each, the record is small: source, author, date, the passage or data that supports the claim, who checked it. That record is what lets a later reader distinguish verified from unexamined, which is the difference between a document that carries evidence and a document that carries formatting.

Two mechanical aids raise the yield. Require generated citations in a form that can be opened — resolvable link or document reference, not a bare title — so that checking is one click instead of a search expedition; unopenable citations get treated as unsupported claims by default. And keep the verification log beside the document rather than in a separate system, because a check that costs a context switch is a check that stops happening in busy weeks.

## Use a named reviewer and escalation path

Review is a control only when a specific person, with relevant competence and protected time, accepts responsibility for the disposition — and has somewhere defined to send what exceeds their authority. Absent any of those properties, review is a signature loop: present, recorded, and useless.

Escalation needs one more property: it must be safe to use. If routing a contested case upward reads as incompetence, reviewers will resolve their doubts locally and quietly — which converts the escalation path into decoration. The forums that receive escalations set this culture in their first few responses, exactly as gate forums do with bad news: thank the reviewer, decide the case, and the path stays alive.

The failure this prevents is nominal approval, which arrives through predictable doors. Volume: the approver clears forty items a day and clicking through becomes the job. Competence mismatch: the reviewer can assess the prose but not the domain claim. Authority mismatch: the reviewer has doubts and no standing to stop the downstream action, so the doubt dies in a comment. Red-team experience makes the general point — finding failure requires people equipped and empowered to look for it [2] — and the same holds for the standing review a workflow runs every day.

The design responses are ordinary management, applied deliberately: match reviewer tiers to consequence tiers from the use declaration; cap review volume before quality collapses, and treat that cap as a capacity constraint on generation itself; separate duties where the requester's interest conflicts with honest review; define escalation for contested or novel cases — a named route, not a mood. And record disposition with reasons, because 'approved, with these caveats' is knowledge the organization will need at the next part of the protocol.

Reviewer fatigue is a design input, not a moral failing. Sustained challenge is cognitively expensive, and a reviewer who has approved two hundred consecutive items has been trained by experience to approve the two hundred and first. Rotation across reviewers, deliberate seeding of known-bad items to keep detection sharp, and honest caps on daily review volume are the same techniques inspection disciplines have used for decades. They apply because this is inspection, whatever the interface looks like.

## Learn from plausible failures

A caught failure is operating evidence. The blended statistic, the miscast policy clause, the calculation with the stale input — each is information about where the workflow's defenses are thin, which prompts and data invite trouble, and what review load is missing what. Kept private, the catch protects one document; fed back, it upgrades the control for everyone.

The failure this prevents is organizational amnesia. Corrections happen in individual sessions — a user notices, fixes, moves on — and the same failure pattern recurs across users and months because nothing connects the catches. The organization pays for the same lesson repeatedly, in the currency of near misses, and its formal confidence in the workflow stays uncalibrated by any of them.

The practice is lightweight capture with real routing: a failure log with category, workflow, detection point, and consequence class; a periodic review that looks for patterns; and authority to change something when a pattern appears — prompts, source data, interface defaults, permitted uses, review tiers, or the evaluation set that tests the system. Near misses count double, because a failure caught by luck is a failure the control missed. The measure of this part working is specific: recurring failure classes should stop recurring.

A starter taxonomy makes the log immediately useful: fabricated support (sources or figures that do not exist as cited); blended fact (real elements combined into a false composite — my steering-deck visitor); stale truth (accurate at training or retrieval time, wrong now); misapplied frame (correct information, wrong jurisdiction, segment, or policy version); and induced agreement (the answer bent to the prompt's visible preference). Five

classes cover most operating incidents, and the class distribution tells the program where to spend: fabrication points at verification discipline, stale truth at data refresh, induced agreement at prompt and interface design.

## The protocol under real conditions

The five parts back each other up, which matters because each is individually escapable. A use declaration without disconfirmation produces well-classified echo; disconfirmation without independent checking breaks framings but not fabrications; independent checking without a real reviewer produces verified inputs to an unaccountable disposition; and all four without learning leave the organization exactly as exposed next quarter. The protocol's strength is the sequence, and its cost discipline is the first part — most output exits at the declaration stage, spending scrutiny only where consequence concentrates it.

EXHIBIT 2

### The disposition each challenge produces

**State the consequential decision**

Apply the protocol when reliance could create material consequence; keep low-risk assistance proportionate.

**Seek disconfirming evidence**

Narrow or withhold the conclusion when it survives only inside one framing.

**Check claims independently**

Remove the claim or reduce reliance when independent evidence cannot be obtained.

**Use a named reviewer and escalation path**

Route high-consequence ambiguity to a qualified owner instead of treating any human touch as sufficient.

**Learn from plausible failures**

Reduce or suspend permitted use when failures outrun the organization's ability to detect and correct them.

Two implementation realities deserve honesty. First, the protocol competes with deadlines, and under pressure the temptation is to skip to the signature. The countermeasure is embedding: the questions live in the interface or work record where reliance happens, not in a policy document nobody opens at 4 p.m. Second, the protocol can be performed — declarations pencil-whipped, disconfirmation prompts run and ignored. The countermeasure is sampling: periodic spot-checks that re-verify a handful of dispositions, with visible consequences when the record and the reality diverge. A protocol occasionally audited stays a protocol; one never audited becomes a form.

The protocol also has a defined relationship with systematic evaluation, and confusing the two wastes both. Evaluation tests the system — representative cases, thresholds, regression after change — and answers whether this workflow should rely on this capability at all. The challenge protocol tests the instance: this output, feeding this decision, today. Evaluation results calibrate the protocol's tiers (a workflow with strong evaluation evidence can verify by sampling rather than exhaustively), and the protocol's failure log feeds evaluation its most valuable cases.

Each control covers the other's blind side: evaluation cannot see the specific convenient number; instance review cannot see the slow drift in aggregate quality.

## Installing the protocol where reliance happens

Installation is deliberately modest:

- Put the five questions into the interface or work record where reliance occurs.
- Define which decisions require independent checking and which can use sampling.
- Calibrate reviewer capacity before increasing generated volume.
- Review failure patterns monthly and change prompts, data, permissions, workflow, and user guidance together.

Embedding looks different by surface, and the details decide adoption. In a document workflow, the use declaration is a template header and the verification log is a table at the end. In a ticketing or CRM flow, the declaration is two required fields and the reviewer disposition is a status. In a chat interface, the disconfirmation pass is a pinned follow-up prompt the team actually uses. None of this needs new software on day one; it needs the questions placed where the work already happens, in the tool people already have open.

Start with one workflow where consequence is real and volume is manageable — contract analysis, customer-facing recommendations, financial summaries — and run the protocol there until the rhythm is boring. The early weeks will surface an uncomfortable number of catches; that is the baseline being discovered. Expand by workflow, not by mandate, letting each new area adapt tiers and sampling to its own consequence profile.

Ownership splits cleanly: workflow owners hold the use declarations and reviewer rosters, the AI program function holds the failure log and the cross-workflow patterns, and the risk function holds the sampling audit. The protocol should not acquire a bureaucracy; three roles and a monthly review carry it.

## What the record shows when it works

A working protocol leaves numbers:

- Output sorted by declared use mode, and the share receiving each review tier.
- Load-bearing claims checked independently, and the rate at which checking changed the claim.
- Reviewer volume against capacity caps, and dispositions with recorded reasons.
- Failures logged by class, detection point, and consequence — including near misses.
- Recurring failure classes retired by a control change.
- Sampled dispositions that survived re-verification.

Resist the temptation to roll these into a compliance percentage. A single 'protocol adherence' score invites exactly the performance the sampling audit exists to catch, and it hides the operationally interesting signals — which claim classes keep failing verification, which reviewers' dispositions keep surviving re-checks, which workflows' declared consequence keeps drifting upward after the fact. The numbers are a diagnostic set, and their disagreements are the findings.

The second measure is the one to watch early: the rate at which independent checking changes claims is the workflow's real error rate, and it is almost always higher than anyone's prior. Watching it fall as prompts, data, and guidance improve is the protocol's return on investment, made visible.

## The strongest counterargument

The protocol does not assume every model answer is false or that a human is automatically right. It creates structured friction where consequence warrants it. Independent evidence, qualified judgment, and recoverable action are the safeguards; generic skepticism is not.

A related objection — that models are improving and the protocol will age out — gets the trend right and the conclusion wrong. Better models shrink the error rate; they do not shrink the consequence of the errors that remain, and they actively grow the plausibility of those errors. A control keyed to consequence rather than to error frequency is exactly the control that survives model improvement: the tiers stay, the verification volume falls as earned evidence accumulates, and the failure log tells you when to relax which check. That is the protocol adjusting, not retiring.

The efficiency objection also deserves a direct answer: yes, the protocol slows the last step of consequential work. That is what it is for. The comparison is not protocol versus frictionless speed; it is protocol versus the incident — the fabricated figure in the board deck, the policy misreading in the customer letter — plus the blanket review mandate that follows the incident and slows everything. Proportionate friction, paid at the moments that matter, is the cheap version of trust.

## The shift

The shift is from treating review as a virtue to running challenge as a control: a declared decision, a real attempt at disconfirmation, claims followed to ground, a named human with the authority and the standing to say no, and failures that teach the system rather than embarrass its users. Fluency stops functioning as evidence, because the workflow no longer accepts it as any.

The confident-and-wrong answer will keep arriving; that is the nature of the tools. What changes is what happens next — whether it meets a reader with a quick eye and a deadline, or a protocol that was built for exactly this visitor.

And there is a compounding return that the incident-avoidance framing undersells. Teams that run the protocol get better at asking: their prompts start carrying disconfirmation requests, their source discipline improves, their instinct for the convenient number sharpens. The control, practiced long enough, migrates into craft — which is what every good control eventually hopes to become.

# Notes and sources

[1] OpenAI, "Sycophancy in GPT-4o: What Happened and What We're Doing About It," April 2025. https://openai.com/index/sycophancy-in-gpt-4o/

[2] Blake Bullwinkel et al., "Lessons From Red Teaming 100 Generative AI Products," Microsoft Research, January 2025. https://www.microsoft.com/en-us/research/publication/lessons-from-red-teaming-100-generative-ai-products/

### About the author

Marco Policani is an enterprise portfolio, PMO, and AI operating-governance leader. He builds portfolio governance systems where AI carries the analytical load and named humans own every decision — an approach documented across case studies, walkthroughs, and working governance guides on his portfolio site.

Portfolio & governance library: policani.net/governance · Full portfolio: policani.net · LinkedIn: linkedin.com/in/marcpolicani