

Auditable Authority: When AI Can Advise, and Who Should Decide

A DAPM Design Companion

By Keith Townsend, <https://thectoadvisor.com>

The Problem No One Can Name

Your AI project is working. The output is good. It's faster than the old process, more consistent, and the team is shipping more than ever.

So why does it feel wrong?

Maybe it's the decision that went out last week that you can't fully explain to the board. Maybe it's the report that's technically accurate but doesn't sound like your organization wrote it. Maybe it's the security policy the AI drafted that's perfectly structured but doesn't reflect how your team actually thinks about risk. Maybe it's the realization that you've been approving AI-generated output without really reviewing it — not because you're negligent, but because it's good enough that reviewing it feels like a waste of time.

You can't point to a failure. That's the problem.

The failure isn't visible because the AI is doing its job well enough that you stopped doing yours. Not all of your job. Just the parts that never had a name — the judgment calls, the institutional voice, the risk posture, the cultural instincts that shaped every decision before AI entered the workflow. Those were never specifications. They were never requirements. They were just *there*, because you and your team were the ones doing the work.

Now the AI is doing the work. And those things are gone. Not because the AI removed them. Because nobody told the AI they existed.

This is happening in every enterprise deploying AI at scale. The projects that fail visibly — hallucinations, bad output, compliance violations — get caught and fixed. The projects that succeed on every metric while silently losing the organization's judgment, voice, and decision-making culture are the ones that do real damage. Because by the time anyone notices, the capacity to notice has itself eroded.

And this isn't new. It's a pattern that repeats every time the compute model changes.

This Happened Before AI

Years ago, a CTO called me to diagnose what he thought was an infrastructure problem. His organization was modernizing off a mainframe. Teams that had worked on a shared system for 25 years were given virtual instances of the software runtime — their own independent environments where they could develop and test their modules individually. The goal was obvious: move faster, ship more, remove the bottleneck of shared infrastructure.

I stopped him before he could finish describing the problem. I asked: instead of reliably shipping faster, did defects increase and delivery slow down?

He asked me how I knew.

It's a classic pattern. The mainframe wasn't just infrastructure. It was a constraint that enforced an integration process — a well-orchestrated sequence of testing and coordination that had evolved naturally over 25 years. Nobody designed it. Nobody documented it. It existed because the shared environment *required* teams to coordinate. The integration knowledge lived in the team, enforced by the constraints of the system they worked on.

The moment they distributed the runtime, that process became a variable. Teams could work independently — which was the point. But the thing that made the old process reliable wasn't the mainframe. It was the institutional orchestration knowledge that the mainframe's constraints had enforced for free. Distribute the compute, and you distribute away the constraint. The knowledge that lived inside that constraint evaporates.

The CTO thought he had an infrastructure problem. He had a hidden decision domain problem. Integration orchestration was an unarticulated property of the process — a constant guaranteed by the old compute model that became an ungoverned variable the moment the compute model changed.

That's the same pattern AI creates today. The compute model changes. Properties that were constants — enforced by the fact that humans did the work — become variables. The organization doesn't know what it lost because it never knew what it had. The failure isn't the new tool. The failure is what the old process guaranteed that nobody named.

This paper is a design tool for preventing that failure. It's a companion to DAPM — the Decision Authority Placement Model — which diagnoses where decision authority is missing or misplaced. DAPM tells you where the gaps are. This companion tells you what to do about them when AI is in the mix.

The core argument is simple:

AI can advise at every layer. But authority must be placed where the outcome can be governed — where the decision path is inspectable, the result is reproducible, and someone can be named when it's wrong.

The answer is not “put a human in the loop.” DAPM already explains why that fails at scale. The answer is to place authority deliberately — sometimes with humans, often with deterministic code or inspectable models, and sometimes by recognizing that the LLM was the wrong tool for the job in the first place.

What follows is the design sequence for doing that.

A note on scope. There are two patterns for introducing AI into an enterprise workflow: replacement, where AI takes over functions a human previously performed, and augmentation, where AI is added alongside the human to enhance their capability. In practice, augmentation is the more successful pattern — precisely because the human remains in the workflow, decision

authority stays more naturally anchored, and the hidden decision domains (culture, voice, institutional judgment) remain present because the person carrying them is still doing the work.

This paper primarily addresses the replacement pattern, which is where most governance failures occur. But augmentation is not immune — and the reason it's not immune is capability-driven drift. As AI output improves, augmentation workflows silently slide into replacement workflows. The human is still nominally present, but they're doing less. The AI's contribution grows. The boundary between “assisting the human” and “doing the work while the human watches” erodes without anyone making that decision. Capability-driven drift is the mechanism by which augmentation becomes replacement — which is why even organizations pursuing the augmentation pattern need the design tool that follows.

The Problem AI Introduces

DAPM's original failure mode is *unplaced authority* — systems that accumulate decision rights through inheritance rather than design.

AI introduces two related but distinct failure modes. Both are specific instances of unplaced authority, but each is harder to detect than the original because AI systems actively produce outputs that look like decisions.

Misrecognized Authority

Misrecognized authority occurs when an organization treats AI-generated output as if it carries decision legitimacy. The system recommends. The organization acts on the recommendation. No one placed the authority.

Unlike traditional unplaced authority — where a platform silently made a tradeoff no one examined — AI systems actively produce outputs that *look like decisions*. They classify. They prioritize. They recommend actions. They generate policy language.

The output format creates the illusion of authority. A fraud score looks like a decision. A generated policy draft looks like an approved policy. A recommended access change looks like an authorized change.

The failure is structural: **capability is being mistaken for authority, and no one has placed the authority deliberately.**

Capability-Driven Drift

Capability-driven drift is the more dangerous failure mode. It occurs not at a single moment of misrecognition, but progressively — as AI output improves and moves further downstream in a workflow without the authority boundaries being updated.

The mechanism: AI is introduced in an advisory role. The model improves. What started as “the AI helps me do my job” becomes “the AI does the job and I review it” becomes “the AI does the job and I approve it.” No one decided to shift from augmentation to replacement. The quality of the output made each step feel like a natural efficiency gain. Intermediate artifacts start being treated as final outputs. The boundary between advisory and authoritative erodes — driven not by any design decision but by the improving quality of the AI’s work.

This is what makes capability-driven drift uniquely dangerous: **the trigger is AI doing its job well**. The output is good enough that the human stops exercising authority — not because they decided to cede it, but because the quality eliminated the felt reason to assert it. Authority is not taken. It is abandoned through the absence of a reason to assert it. And the abandonment is invisible because the output quality masks it.

What drifts first — and furthest — is culture: the unarticulated principles that constrain how an organization makes decisions in ambiguity. A professional services firm that operates on “we don’t do things just for money” is applying a cultural constraint that sits above any financial analysis. A firm that values “navigating the gray” is asserting that ambiguity is where their judgment matters most. An LLM can evaluate profitability and assess risk. But it can’t *mean* it — it doesn’t have the cultural context to know which engagements to walk away from, which gray areas to navigate toward and which to navigate away from, which technically-correct decisions feel wrong to the people who built the institution.

Cultural drift is self-reinforcing. Once an organization starts ceding cultural decisions to AI, the people who used to make those decisions stop exercising that judgment. The institutional memory of “how we do things” is replaced by “how the AI does things.” Over time, the organization can’t articulate what it lost — because the culture that would have recognized the loss has itself been eroded.

Governance checklists do not catch this because they assume the human *wants* to maintain authority. The actual failure is that high-quality AI output creates structural pressure to *not* maintain authority — and the authority most likely to be ceded is cultural authority, because it’s the hardest to name, the hardest to measure, and the easiest to let go when the output looks good.

Design Step 1: Do You Need This Compute Model?

Before addressing where authority should be placed, the design tool asks a prior question:

Is an LLM the right compute model for this problem?

Not “can an LLM do this?” — it probably can. The question is whether it should. This is a fitness-for-purpose assessment, not a capability assessment.

Many problems being handed to LLMs are already solved by deterministic compute. Document parsing and form filling is string matching, regex, and template mapping. It runs fast, cheap, and

produces the same output every time. An LLM does the same work at higher cost, slower, with the possibility of hallucinating a field value — and you can't explain why it filled a particular value other than "the model thought so."

Many other problems are well served by inspectable models that already exist. A credit score is produced by a model with well-defined weights. The decision path is transparent. The inputs are known. The output is reproducible. When a regulator asks why an applicant was denied, the institution can trace from inputs through weights to output. The decision is auditable because the model is inspectable.

Replace that inspectable model with an LLM and the analytical capability may improve — the LLM can ingest more signals, find nonlinear patterns, assess risk across a broader feature space. But the decision path becomes opaque. A model with 7 billion parameters is not inspectable the way a logistic regression is. A black-box model with a trillion parameters behind an API is even less so. If there's a correlation with protected characteristics embedded somewhere in those parameters, you can't demonstrate that there isn't. The marginal improvement in risk assessment may not be worth the regulatory exposure you introduce by losing inspectability.

The smarter design: don't replace the inspectable model. Use the LLM to *improve* it. The LLM analyzes gaps in the existing scoring model — finds the patterns the logistic regression misses, identifies signals it doesn't weight, surfaces edge cases where the current model underperforms. Those findings feed back into the inspectable model as new features, adjusted weights, or refined criteria. The inspectable model gets better. The decision path stays transparent. The LLM contributed intelligence without ever touching the decision.

That's the design tool applied to its own logic. The LLM is advisory. The inspectable model holds authority. The decision path remains auditable. You get the analytical benefit without the inspectability tradeoff.

The design sequence begins here:

Is this a deterministic problem? If the problem can be solved with deterministic compute — parsing, matching, rule evaluation, template application — use deterministic compute. The authority question doesn't arise because the system is inspectable and reproducible by design. The LLM adds cost, latency, and non-inspectability to a problem that didn't need any of those.

Is there an inspectable model that solves this? If the domain already has a working model with known weights, transparent decision paths, and reproducible outputs, don't replace it. Use the LLM to find the gaps in the existing model — the patterns it misses, the signals it underweights, the edge cases it handles poorly. Feed those insights back into the inspectable model. The model improves. The decision path stays defensible. In domains where the path must be traceable — lending, insurance underwriting, clinical risk scoring — this is the only design that preserves both analytical improvement and inspectability.

Where do the non-deterministic edges live? If the problem has both deterministic components and genuine ambiguity, scope the LLM to the edges — the exceptions, the novel

patterns, the cases where existing models and rules don't cover the territory. The LLM handles what only it can handle. Deterministic compute and inspectable models handle the rest. Don't hand the entire workflow to an LLM because it can handle the entire workflow.

This upstream gate eliminates a large category of AI governance problems by not creating them. An LLM that was never introduced into a deterministic workflow doesn't need authority boundaries. An inspectable model that retains its role doesn't need a human-in-the-loop workaround. The cheapest governance is the governance you don't need because you chose the right tool.

Inspectability as the Governing Property

The original version of this design tool used “probabilistic” as the property that limited where AI could hold authority. That's imprecise. Plenty of probabilistic systems hold authority in production — and they should.

A credit scoring model is probabilistic. It produces a score based on statistical weights. But it holds decision authority over loan approvals, credit limits, and interest rates across the entire financial system. This works because the model is **inspectable**: the weights are known, the inputs are defined, the decision path can be reconstructed, and the output is reproducible given the same inputs.

The property that determines whether a system can hold auditable authority is not whether it's probabilistic. It's whether the decision path is inspectable.

Inspectability means: can you trace from inputs through the decision process to the output? Can you explain *why* this input produced this output? Can you reproduce the result? Can you demonstrate that the decision path does not depend on factors it shouldn't?

Inspectable systems — whether deterministic code or statistical models with known weights — can hold authority in auditable domains because their decisions can survive scrutiny. Non-inspectable systems — including LLMs of any size — cannot hold authority in those domains, not because they produce worse outcomes, but because the decision path is opaque.

This is the governing property for the entire design tool:

As inspectability requirements increase, authority must migrate away from non-inspectable systems and toward systems whose decision paths can be traced, reproduced, and defended.

This does not exclude AI from the decision space. It constrains where AI can hold *authority* based on whether the domain requires the decision path to be reconstructable. In domains where it doesn't — exploratory analysis, draft generation, pattern discovery — AI can hold authority directly. In domains where it does — regulated decisions, legal exposure, compliance

enforcement — authority must reside in a system whose reasoning can be opened and examined.

Design Step 2: Surface the Hidden Decision Domains

When AI enters a workflow, it doesn't just automate existing decisions. It converts properties that were previously constants of the process into variables — and each of those variables becomes a decision domain that requires explicit authority placement.

This is the step most organizations skip, and it's why AI projects fail in ways that no one anticipated.

Start with culture — which is always a hidden decision domain when AI enters a workflow. The mechanism and the danger were established earlier: culture is the first authority ceded, the hardest to measure, and the most self-reinforcing when it erodes. It doesn't need to be discovered. It needs to be assumed.

But culture is not the only hidden domain. In any human-operated workflow, other properties are guaranteed by the process itself:

A security team writes security policy. The risk posture and policy intent are constants — not because someone specified them, but because the people who understand the organization's risk tolerance are the ones writing.

An analyst assembles a financial report. The framing, the narrative emphasis, the way the numbers are contextualized are constants — because the analyst understands the audience's perspective.

A content creator produces editorial work. The voice, the perspective, the institutional identity are constants — because the work is written by someone who *is* the institution.

The moment AI enters any of these workflows, every one of those constants becomes a variable. The AI produces output that may be excellent on every measurable dimension while silently diverging on dimensions that were never measured because they were never at risk.

The Design Step

Before introducing AI into any workflow, enumerate the properties of the current process that are guaranteed by human execution. Start with culture. Each property becomes a new decision domain the moment AI enters the workflow. Each one requires explicit authority placement.

This enumeration is not intuitive. The properties that are most at risk are precisely the ones that are hardest to name, because they've never been named. They were invisible features of the process, not explicit requirements. Culture was never a specification. Voice was never a

parameter. Institutional perspective was never a deliverable. They existed because the process produced them as byproducts of human involvement.

The design tool requires making them explicit. For each property identified:

Name it. What was the constant? Culture. Voice. Framing. Risk posture. Policy intent. Institutional perspective. Contextual judgment. Whatever the human brought to the process that wasn't captured in any requirements document.

Determine its inspectability requirement. Is this a property that must be auditable? Culture in a regulated communication or client-facing engagement has high inspectability requirements — the output must reflect the organization's principles, and the organization must be able to verify that it does. Internal draft framing may be low.

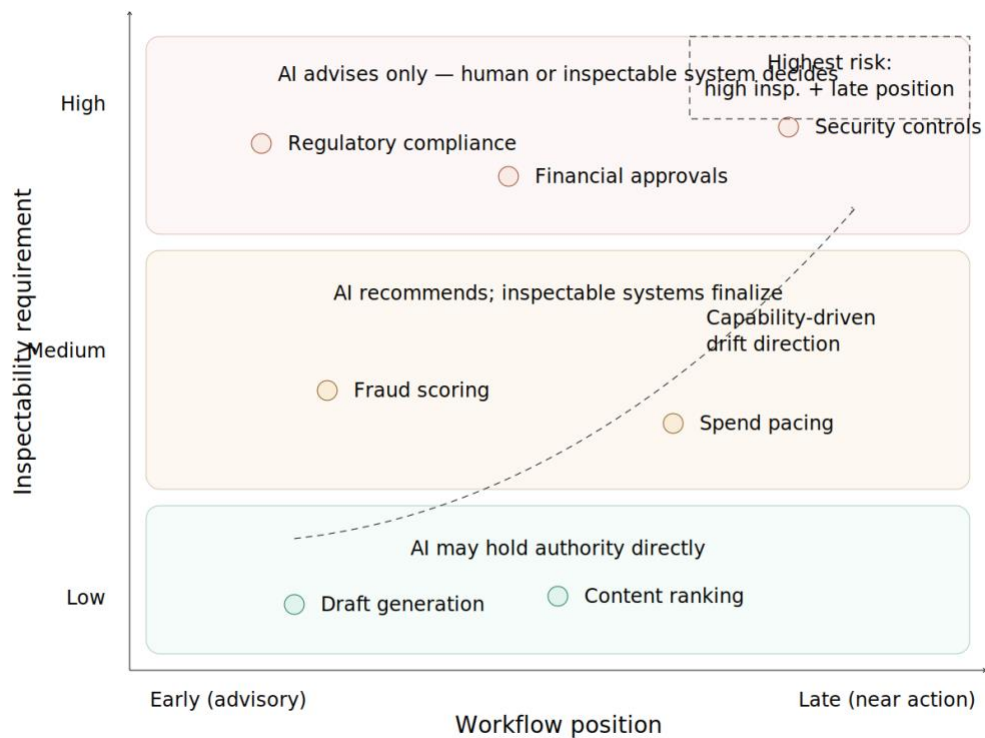
Place authority over it explicitly. Who holds authority over culture? Over voice? Over framing? If the answer is "the AI will handle it," that's an authority placement — make it deliberate, and understand that for cultural properties, it is almost certainly the wrong one. If the answer is "the human retains it," define how. If the answer is "it doesn't matter for this context," that's also a valid placement — but it must be explicit, not assumed.

Properties that are not enumerated will not be governed. Authority over them will be unplaced. And because these properties were invisible before automation, the drift will be invisible too — surfacing only in downstream metrics that no one connects back to the authority gap, because no one knew the authority gap existed.

This is the structural reason enterprise AI projects fail in ways that surprise the people who built them. The system works. The output is good. And something is wrong that nobody can name — because the thing that went wrong was never a named property of the process to begin with.

Design Step 3: Inspectability Gradient

The inspectability gradient is a placement heuristic. It maps two dimensions — the inspectability requirements of a decision domain (including the hidden domains surfaced in Step 2) and the workflow position of AI output — to viable authority placements.



Domain Requirements

The first dimension is what the decision domain requires for its outcomes to be governed.

Low inspectability requirements. AI may hold authority directly. Outcomes do not require traceable decision paths, reproducibility, or external defense. Examples: internal content ranking, draft generation, noncritical routing, exploratory analysis.

Medium inspectability requirements. AI recommends; inspectable systems finalize within guardrails. Outcomes require consistency and evidence but operate within bounded parameters where deterministic code or inspectable models can enforce policy. Examples: fraud scoring with threshold-based disposition, spend pacing within approved budgets, automated service tier assignment.

High inspectability requirements. AI advises only. Authority is placed with humans or inspectable systems — whichever can produce outcomes that are traceable, reproducible, attributable, and defensible. Examples: security control enforcement, financial approvals, access management, regulatory compliance workflows, patient-impacting clinical decisions.

The gradient is not a maturity model. Low-inspectability domains are not immature — they are domains where the consequences of a wrong decision do not require a defensible decision path.

Note that hidden decision domains surfaced in Step 2 must be individually assessed on this gradient. A workflow may have low inspectability requirements for its primary output but high requirements for a previously-hidden property. Content generation may be low-inspectability for structure and completeness but high-inspectability for voice attribution. If the hidden properties are not assessed separately, authority over them defaults to unplaced.

Workflow Position

The second dimension is where AI output lands in the workflow. This matters because workflow position determines de facto authority regardless of intent.

AI output that sits early in a pipeline — feeding into human review, further analysis, or deterministic processing — is structurally advisory. The system has downstream checkpoints that exercise authority over the final outcome.

AI output that sits late in a pipeline — close to or at the point of action — exercises de facto authority whether anyone grants it or not. If the AI's recommendation is the last substantive input before a decision is executed, the AI is functionally the authority holder even if the org chart says otherwise.

The design rule: **the closer AI output sits to the point of action, the stronger the authority boundary must be.** Workflow position is a governance concern, not just an architecture concern. When AI output migrates downstream — through capability improvement, workflow optimization, or simple convenience — authority boundaries must be re-examined.

This is where capability-driven drift becomes operationally visible. The AI's output quality improves. The output moves further downstream. The boundary erodes. The design tool must define not just where authority is placed, but where in the workflow AI output is permitted to land — and what happens when it migrates past that boundary.

Why Human-in-the-Loop Is Not the Answer

The default industry response to AI governance concerns is human-in-the-loop. DAPM already explains why this fails as a general solution.

Governance-coupled authority — which is what human-in-the-loop operationally produces — is structurally non-viable in complex domains. When decision frequency increases and causality becomes nonlinear, human escalation cannot keep pace. The result is either rubber-stamping (humans approve without examining, making the oversight theater) or bottlenecking (humans slow the system to a pace incompatible with operational requirements).

Neither of these is governance. One is the appearance of governance with no substance. The other is governance that destroys the value automation was supposed to provide.

Capability-driven drift makes this worse. Even when human-in-the-loop is architecturally present, high-quality AI output undermines it operationally. The human is in the loop. The human reviews. But the review becomes cursory because the output looks good — and over time, the review degrades into the rubber-stamping failure mode not because the human is negligent, but because the AI's quality made genuine scrutiny feel unnecessary.

Human-in-the-loop also fails specifically for hidden decision domains. A human reviewer checking for output quality will not catch voice drift, framing shifts, or risk posture migration — because those properties were never part of the review criteria. The human is in the loop for the *visible* decision domains. The hidden ones pass through unexamined. Adding a human to the loop does not help if the human is not reviewing for the properties that are actually at risk.

The design tool offers a different framing. The question is not *whether a human should review*. The question is *what the decision domain requires for its outcomes to be governed*. If the requirement is a traceable decision path, an inspectable model or deterministic code may be a better authority locus than a human reviewer. If the requirement is judgment in the face of ambiguity, a human is the right placement — but that's a design decision about the domain, not a blanket policy about AI.

Solving for inspectability and auditability instead of solving for human comfort produces authority placements that actually survive operational reality.

The Firewall Test

A practical diagnostic for authority placement:

Would you allow an AI to silently create and enforce a new firewall rule?

Most enterprises answer no. But the interesting question is *why* they answer no.

It's not because AI can't analyze network traffic effectively. It can. It's not because the AI would necessarily produce bad rules. It might produce excellent ones.

The answer is no because what gets audited in that domain is not intelligence. It's approved policy, enforcement consistency, change records, evidence trails, and accountable owners. Each of those requires an inspectable decision path.

The AI can advise on all of those. But *holding authority* over firewall policy requires producing outcomes that satisfy each of those audit requirements. Non-inspectable systems cannot do that — not because they lack capability, but because their decision paths cannot be traced, reproduced, or defended.

The firewall test generalizes. For any decision domain, ask: *what does this domain audit?* If the audit requires a traceable decision path — evidence, reproducibility, attribution, defensibility — authority cannot rest in a system whose reasoning is opaque.

Placement Targets

When the upstream gate confirms that an LLM is the right compute model for a given function, and the inspectability gradient indicates that it cannot hold authority, two placement targets remain. But placement is not a blanket assignment across an entire domain — it is context-dependent within a domain.

Human Authority

Place authority with humans when the decision requires judgment that cannot be codified, when exceptions dominate rules, when ethical considerations are primary, when accountability must be personal and nameable, when ambiguity is irreducible, or when a hidden decision domain requires human judgment to preserve (voice, institutional perspective, contextual framing).

Inspectable System Authority

Place authority with inspectable systems — deterministic code or inspectable models — when policies are explicit and codifiable, when repeatability is a requirement, when decision paths must be traceable, when decision tempo exceeds human review capacity, or when control enforcement must be consistent and demonstrable.

AI Advisory Role

AI remains in the system — not as an authority holder, but as an advisory function. AI contributes analysis, forecasting, pattern recognition, summarization, prioritization, scenario generation, and risk scoring.

The critical design constraint: **AI output feeds into authority functions but does not constitute authority itself.**

Context-Dependent Authority Grants

Authority placement is not one decision per domain. Within the same workflow, AI may hold different levels of influence depending on the specific output context — and the hidden decision domains within that context.

Consider a content production workflow. AI may be granted significant co-authoring authority for formal documents where structure, completeness, and consistency are the primary requirements — whitepapers, case studies, compliance documentation. In the same workflow, AI may be constrained to advisory-only for outputs where voice, perspective, and institutional identity are primary — editorial content, public positions, thought leadership.

The domain is the same: content production. The inspectability requirement differs by context. Formal documents are audited for completeness and accuracy — domains where AI co-authoring is structurally viable. Editorial content is audited for attribution and voice — domains where AI authority produces the quality-driven cession failure.

The difference between these contexts only becomes visible if Step 2 has been completed — if the hidden decision domains (voice, perspective, institutional identity) have been surfaced and their inspectability requirements assessed. Without that enumeration, the entire content production workflow receives a single authority placement, and the hidden domains drift.

Handoff Design: Where Failures Actually Occur

Once authority is placed, the system must manage transitions between AI advisory functions and authority-holding functions. These handoffs — not the AI model and not the authority holder — are where most operational failures occur.

A critical design requirement for every handoff: **the handoff must review for the properties that are actually at risk — including the hidden decision domains surfaced in Step 2, not just the visible output characteristics.** A handoff that checks for quality but not for voice is a handoff that will pass every time while drift accumulates undetected.

AI → Human review. AI generates a recommendation. A human evaluates and decides. The failure mode is rubber-stamping — compounded by capability-driven drift, where improving AI quality makes rubber-stamping feel rational. Design response: the handoff must include the basis for the recommendation, not just the recommendation itself. The human must be able to evaluate *why*, not just *what*. And the review criteria must explicitly include the hidden decision domains.

AI → Inspectable system. AI produces a recommendation. Deterministic code or an inspectable model evaluates against policy and executes. The failure mode is scope creep: the inspectable system enforces beyond its authorized domain because the AI recommendation was broader than the policy was designed to handle. Design response: explicit boundary definition — what the inspectable system is authorized to enforce and what must escalate.

Inspectable system → Human escalation. The policy engine or inspectable model encounters a condition outside its rules. The failure mode is escalation flooding: too many exceptions reach human reviewers, collapsing back into governance-coupled authority. Design response: the escalation threshold is a design parameter, not an afterthought. If the exception rate is too high, the policy needs revision — the answer is not more human reviewers.

Human → AI re-analysis. A human decision generates new data. AI incorporates it for future recommendations. The failure mode is feedback drift: the AI learns from human decisions that were themselves compromises or errors, producing increasingly skewed recommendations. Design response: human decisions that feed back into AI advisory systems must be distinguished from validated ground truth.

Handoff design is not an add-on. It is a core component of the authority placement. An authority placement that does not specify how transitions work between advisory and authoritative functions is incomplete — and will fail at the seams.

Mapping to the DAPM Authority Matrix

This design tool does not replace the DAPM Authority Matrix. It extends the matrix by adding columns for Advisory Source, Inspectability Requirement, and Compute Model.

Decision Domain	Compute Model	Authority Placement	Advisory Source	Accountability	Inspect. Req.
Cost / Spend Control	Deterministic + LLM (edges)	Code (policy engine)	AI (forecasting)	Finance / Exec	Medium
Security Enforcement	Deterministic	Code + Human (exceptions)	AI (threat detection)	Risk / CISO	High
Compliance / Audit	Deterministic + Human	Human (review)	AI (doc analysis)	Legal / Compliance	High
Credit Decisioning	Inspectable model	Model + Code	N/A — model holds authority	Risk / Compliance	High
Performance & Scaling	Deterministic + LLM	Code (autoscaler)	AI (demand prediction)	Product Team	Low–Medium
Access Management	Deterministic	Code + Human (exceptions)	AI (risk scoring)	Security / Risk	High
Document Processing	Deterministic (LLM edges)	Code (parser)	AI (ambiguous fields)	Operations	Low–Medium

Credit decisioning appears in this matrix specifically to demonstrate that inspectable models can and should hold authority when the domain requires a traceable decision path. Not every AI governance problem requires an LLM governance solution. Some require keeping the LLM out.

Note that hidden decision domains surfaced in Step 2 would appear as additional rows in an organization's authority matrix. Voice authority in a content workflow, framing authority in financial reporting, policy intent authority in security documentation — each surfaces as its own row with its own compute model, authority placement, and inspectability requirement.

Relationship to Layer 2C and the Reasoning Plane

The Reasoning Plane (Layer 2C in the 4+1 AI Infrastructure Model) is the architectural layer where advisory intelligence operates. It handles context management, reasoning orchestration, and the coordination of AI capabilities that inform decisions.

This design tool governs what happens at the *boundary* of Layer 2C — the point where advisory output exits the Reasoning Plane and enters a domain where authority must be placed.

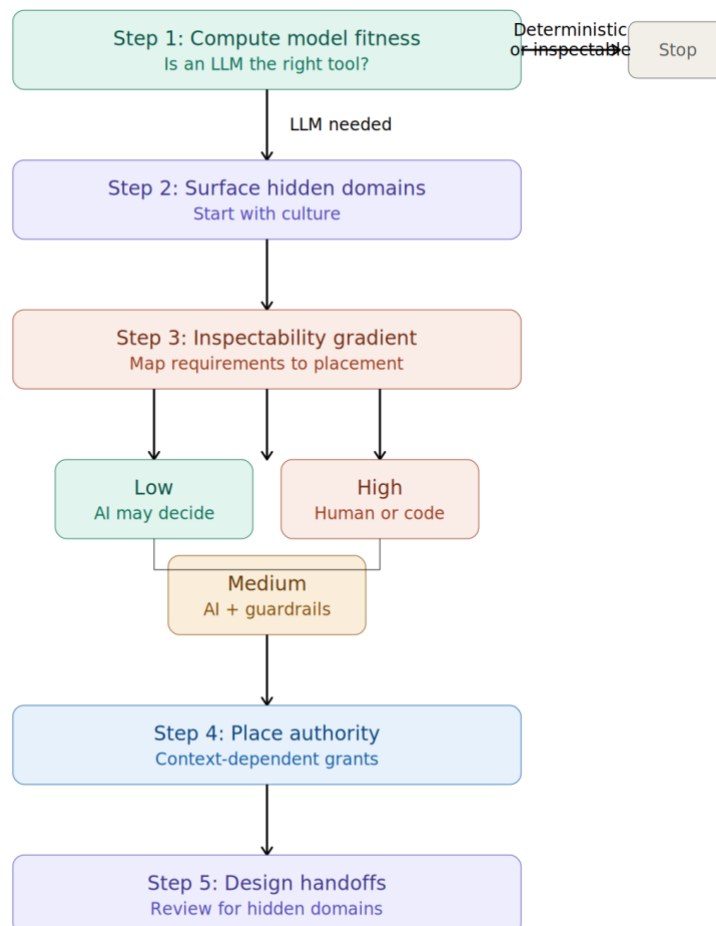
Layer 2C produces intelligence. This design tool determines where authority over that intelligence resides. The two are complementary: Layer 2C defines the *architecture* of advisory systems; this tool defines the *governance* of authority placement around those systems.

The upstream compute model gate also intersects with Layer 2C. Not every function within an enterprise workflow needs to pass through the Reasoning Plane. Deterministic functions and inspectable models operate outside Layer 2C entirely. The design tool identifies which functions benefit from reasoning-layer intelligence and which are better served by compute models that never enter that layer.

Without this governance layer, Layer 2C outputs flow into enterprise systems with no explicit authority placement — reproducing the misrecognized authority problem at architectural scale. And without explicit workflow position constraints, those outputs migrate downstream through capability-driven drift until they exercise de facto authority over decisions they were designed to advise.

The Design Sequence

The complete design sequence, summarized:



Step 1: Compute model fitness. Is an LLM the right tool? If the problem is deterministic, use deterministic compute. If an inspectable model already solves it, keep the inspectable model. Scope the LLM to the edges where it's genuinely needed. Don't create governance problems that don't need to exist.

Step 2: Surface hidden decision domains. Enumerate the properties of the current process that are guaranteed by human execution. Start with culture — the unarticulated principles that constrain how the organization makes decisions in ambiguity. Culture is always a hidden decision domain when AI enters a workflow. Then identify other constants: voice, framing, risk posture, institutional perspective. Each becomes a variable — and a new decision domain — the moment AI enters the workflow. Name them. Assess their inspectability requirements. Properties that are not enumerated will not be governed.

Step 3: Inspectability gradient. For each decision domain — including the hidden ones — map inspectability requirements and workflow position to viable authority placements. High-inspectability domains place authority with humans or inspectable systems. Low-inspectability domains may allow AI authority directly.

Step 4: Authority placement. Place authority explicitly — with humans, with inspectable systems, or with AI where the domain permits. Use context-dependent grants where different output types within the same workflow have different inspectability requirements.

Step 5: Handoff design. Specify how transitions work between advisory and authoritative functions. Define review criteria that include hidden decision domains, not just visible output quality. An authority placement without handoff design is incomplete.

Implications

Organizations adopting AI should stop asking “can AI do this?” and start asking:

Do we need AI for this? If the problem is solved by deterministic compute or an inspectable model, the AI governance question doesn't arise. Use the tool that fits. The cheapest governance is the governance you don't need because you chose the right compute model.

What becomes a variable when AI enters this workflow? Every property that was previously guaranteed by human execution becomes a decision domain requiring explicit authority placement. If you can't enumerate what changes, you can't govern what drifts. The things most likely to drift are the things you've never had to name — because the process itself used to guarantee them.

Is the decision path inspectable? If the domain requires a traceable, reproducible, defensible decision path — because of regulation, legal exposure, or the consequences of being unable to explain — and the model is non-inspectable, the answer isn't to architect around the limitation. The answer is to use a different model. The marginal improvement in capability is not worth the inspectability you lose.

Where does AI output land in the workflow? If AI output sits at or near the point of action, it exercises de facto authority regardless of organizational intent. The closer AI output sits to the final decision, the more explicit the authority boundary must be — and the more actively the organization must resist the quality-driven pressure to let that boundary erode.

Can we explain this enforcement? If enforcement must be explained to an auditor, a regulator, or a court, the enforcing system must produce traceable, reproducible outputs. AI can inform the enforcement decision. It cannot be the enforcement mechanism.

Who is accountable when this decision is wrong? If no one can be named, authority has not been placed. DAPM's original diagnostic applies. This design tool then determines where to place it given the domain's inspectability requirements.

Is the AI getting better at this? If yes, the authority boundary is under pressure. Model improvement is the primary driver of capability-driven drift. Every capability upgrade should trigger a review of where AI output sits in the workflow and whether the authority boundaries still hold.

Thesis

AI expands the decision space. It brings intelligence to domains where analysis was previously unavailable, too slow, or too expensive.

But intelligence is not authority. Capability is not legitimacy. And the ability to produce an answer is not the same as being authorized to act on it.

The greatest risk is not that AI produces bad output. It is that AI produces good output — good enough that organizations stop exercising the authority they never formally ceded. The drift is invisible because the quality masks it. The failure surfaces only in downstream metrics, by which point the authority boundary has already eroded.

The most common mistake is not misplacing authority. It is reaching for an LLM when the problem was already solved — by deterministic code, by an inspectable model, by a system whose decision path could be traced and defended. The governance problem didn't need to exist. It was created by choosing the wrong tool.

And the most dangerous mistake is ceding culture. As models improve, organizations gradually hand over the decisions that define who they are — how they navigate ambiguity, what principles constrain their choices, what makes their judgment recognizably theirs. These are the decisions AI is least equipped to hold and most likely to be given, because they're the hardest to specify, the hardest to measure, and the easiest to let go when the output looks good. Once cultural authority drifts, the organization loses the capacity to recognize what it lost — because the culture that would have caught the drift has itself been eroded.

DAPM diagnoses where authority is missing or misplaced. This companion provides the design sequence for resolving it:

First: choose the right compute model. Don't create governance problems you don't need.

Then: surface what changes. Start with culture. Name the properties that were constants and are now variables.

Then: place authority deliberately — over every decision domain, visible and hidden — where the decision path can be inspected and the outcome can be governed.

The question was never “should we put a human in the loop?”

The question is: **can the outcome be governed?**

Place authority there.