10

WHITE PAPER · WORK, ADOPTION & JUDGMENT

The Coordination Tax

Design the coordination topology before declaring work automated

Marco Policani

Enterprise Portfolio · PMO · AI Operating Governance

policani.net · linkedin.com/in/marcpolicani

July 2026

EXECUTIVE SUMMARY

Automation changes where coordination happens; leaders must design the handoffs, exceptions, and evidence that remain.

Automation can collapse a task while the work between tasks — routing, translation, waiting, approval, reconciliation, repair — absorbs the savings invisibly. This paper maps that coordination topology: observed cases traced end to end, every link binned by value, boundary work priced at the seniority that performs it, and released capacity counted only when the whole path confirms it.

01 Trace observed cases, not the process diagram — the tax lives on the edges the diagram does not show.

02 Bin every coordination link before deleting any: programs that treat the whole category as waste delete their own immune system.

03 Count released capacity at the path level after the topology settles, or staff for fictional savings and run hot for a year.

The argument

Automation can remove task effort while leaving — or increasing — the work required to route, reconcile, approve, explain, and recover. Leaders should map the coordination topology and measure the total workflow before declaring productivity or autonomy.

The automation review that made this vivid for me involved a report-drafting workflow that had, by its own metrics, succeeded completely. Drafting time per report had fallen from four hours to twenty minutes. The team was proud, the tooling was good, and the function's total throughput had not improved at all. We traced a month of reports to find out where the fourteen weekly hours of savings had gone. The answer was everywhere and nowhere: analysts now spent time selecting which source data the tool should use, correcting its formatting for the downstream system, explaining in review meetings which parts were generated, waiting on an approval step that had been added because the reports were generated, and reconciling discrepancies between the tool's numbers and the ledger's. The task had collapsed. The coordination around it had metastasized.

That is the coordination tax: the work between the tasks — routing, translation, waiting, approval, reconciliation, explanation, repair — which rarely appears in an automation business case because it is distributed across many roles, buried inside calendars and inboxes, and owned by no one in particular. Task automation concentrates its benefit in one visible place and scatters its costs across many invisible ones. Unless the surrounding topology is mapped and measured, the business case compares a real saving against an unpriced expense.

This paper offers the map as the instrument: a coordination topology review with six disciplines, run before scale decisions. Model-based research on when coordination between tasks is genuinely avoidable offers a useful hypothesis about how AI may change work; NIST's risk framework supplies the more durable governance frame [1] [2]. Neither substitutes for measuring one's own workflow, which is the entire method here — one I assembled by

tracing my own automations' missing savings. It is written for operating executives, transformation leaders, workflow owners, and AI program leaders.

Why the tax stays invisible

Coordination work has three properties that keep it off the books. It is fragmented: thirty seconds of routing here, five minutes of reconciliation there, no fragment large enough to name. It is distributed: each role carries a sliver, so no single person experiences the total. And it is elastic: capable people absorb it into their days without complaint, which means it registers as busyness rather than as cost. Work with those three properties does not appear in time studies built around tasks, and automation cases are built around tasks.

Automation then adds its own species to the ecosystem. Generated output must be checked against sources, translated into downstream formats, and explained to consumers who need to know what to trust. New approval steps appear precisely because the work is machine-produced. Exception volume rises at the edges of the automated envelope. None of this is hypothetical or rare — it is the ordinary physics of inserting a fast, narrow capability into a broad, human process. The question is never whether the surrounding coordination changes; it is whether anyone is watching it change.

The strategic risk compounds at portfolio level. Ten automations, each locally successful by task metrics, can sum to a function that is no faster, more brittle, and harder to staff — because each automation quietly recruited the same experienced people into checking, translating, and repairing roles nobody designed or counted. The portfolio celebrates ten wins and wonders why the outcome curve is flat. The answer is in the topology, which no one drew.

Agentic automation raises the stakes on all of this rather than escaping it. An agent that plans, calls tools, and hands work to other agents is a coordination topology in miniature — with the same handoff contracts, the same exception edges, and the same translation layers, now executing at machine speed without the human elasticity that used to absorb design gaps quietly. The organizations that mapped their human topologies will recognize every failure mode; the ones that never learned to see coordination will meet it at a much less forgiving tempo [2].

The coordination topology map

The map is a review with six disciplines, applied to one workflow at a time. It treats the path — not the task — as the unit of analysis, sorts the coordination it finds by value, and puts the results into the same business case that proposed the automation. Nothing in it requires new tooling; it requires observation, which is cheaper and rarer.

EXHIBIT 1

Six disciplines expose where automation's savings actually went

1. **Map work between tasks**
   The unit of analysis is the path across people, systems, decisions, and queues—not the isolated activity the tool performs.
2. **Distinguish productive coordination**
   Some interaction creates judgment, learning, trust, safety, or necessary integration; eliminating it can degrade the outcome.

3. **Find boundary work**
   Automation often creates new translation between generated output and the systems or people that must use it.
4. **Design handoffs deliberately**
   Every transfer needs a clear object, receiving condition, owner, and response expectation.
5. **Own exceptions and recovery**
   A faster happy path can increase the absolute number or complexity of cases requiring human coordination.
6. **Measure the whole outcome**
   Value should appear in end-to-end cycle time, quality, capacity, coverage, decision speed, or risk; tool usage and task duration are supporting measures.

Map work between tasks

The first discipline traces real cases across people, systems, decisions, and queues, from the demand's arrival to the outcome's delivery. Every transfer, wait, approval, dependency, and backward loop gets marked. The tracing must follow observed cases, not the process diagram — the diagram shows the intended path, and the tax lives on the actual one, in the steps people quietly added to make the intended path survive contact with reality — steps that, almost by definition, no document mentions and no owner defends.

The evidence is a touch-and-queue ledger per case: how many hands, how much waiting between hands, how much rework, which role carried each step, and why cases moved backward when they did. A dozen traced cases usually suffice to expose the shape. The rule that follows: no automation decision until the proposal can state, on this map, which links it removes, which it relocates, and which it creates. 'It makes the drafting step faster' is an answer about a node; the tax lives on the edges.

What tracing actually looks like, for concreteness: pick twelve recent cases spanning routine and messy, and for each one interview the people who touched it — not about the process, about this case. When did it arrive on your desk, what did you need before you could act, whom did you wait on, what did you send back, and why? Plot the twelve journeys on one wall. The wall will disagree with the process diagram in the same three or four places for most cases, and those disagreements are the standing coordination structures nobody chartered — the shadow steps the organization built because the official path did not work.

Distinguish productive coordination

The second discipline resists the efficiency reflex. Some interaction is the mechanism by which judgment forms, defects surface, novices learn, and cross-functional risk gets caught: the review that catches the mispriced contract, the handoff conversation where the receiving team hears the case's history, the escalation that pulls a senior eye onto an unusual pattern. Deleting those links produces a faster workflow that fails more expensively.

The test for each coordination point is what it produces: a decision, a defect catch, knowledge transfer, a control, a relationship the work later depends on — or nothing but motion. The honest answers sort links into four bins: valuable (keep, maybe strengthen), necessary but reducible (lighter mechanism, same function), avoidable (route or rule can remove it), and created by the intervention itself (a cost of the automation, to be counted as one). The

bins matter because automation vendors and efficiency programs implicitly treat the whole category as waste, and organizations that believe them delete their own immune system.

A binning example shows the judgment involved. In an order-exception workflow I reviewed, a daily cross-team standup looked like a textbook deletion candidate — thirty minutes, eight people, mostly status. Traced against cases, it turned out to be the only point where pricing learned about inventory substitutions before customers did; deleting it would have converted a coordination cost into a refund stream. The standup stayed, shrank to twelve minutes, and gained an agenda. Meanwhile a weekly 'alignment review' that everyone defended traced to zero decisions and zero catches across a month of cases, and died unmourned. The bins are not discovered in the org chart; they are discovered in the cases.

Find boundary work

The third discipline hunts the tax's newest species: the translation layer that grows where generated output meets the systems and people that must use it. Users normalize formats, re-key content into the system of record, verify claims against sources, annotate provenance for downstream consumers, and prepare inputs so the tool can handle them. This work is nearly always invisible — it happens inside the users' existing roles, in minutes-per-case quantities, at exactly the boundary the task metric does not cover.

The method is observation: sit with real users, time the before-and-after work around the tool, and categorize it — input preparation, output correction, format translation, verification, explanation. The findings convert directly into design: translation dominating a boundary argues for interface or integration work; verification dominating argues for provenance features or upstream quality; explanation dominating argues that downstream consumers need different packaging. Boundary work that cannot be designed away belongs in the business case as a permanent operating cost, priced at the seniority of whoever performs it — which is often uncomfortably high.

Boundary work also has a seniority gradient that compounds its cost. The checking and translation land disproportionately on experienced staff, because judgment is what checking requires — which means the automation's hidden payroll is drawn from exactly the people the organization can least spare, while the visible savings were priced at the average. A boundary-work ledger that records who performs the minutes, not just how many, regularly reverses the sign of a business case.

Design handoffs deliberately

The fourth discipline engineers the transfers that remain. A handoff is a small contract: an object (with its evidence attached), a receiving condition (what state makes it acceptable), an owner on each side, and a response expectation. Automation strains these contracts by changing volume and shape at machine speed — an accelerated upstream step can flood a downstream queue that lacks the context, capacity, or authority to act on what arrives.

A handoff example from the same order-exception review: the automated classifier accelerated case intake threefold, into a resolution team whose acceptance criteria had never been written down. Cases arrived missing the two fields resolvers actually needed, clarification requests tripled, and the 'faster' workflow's end-to-end time got worse. The fix was a handoff contract — two required context fields, an acceptance check at the queue's

mouth, and an intake rate matched to resolution capacity — none of which touched the classifier. The automation was fine. The edge was undesigned.

The evidence is queue telemetry at each transfer: age, acceptance rate, clarification requests, incomplete handoffs bounced back, downstream idle-then-overloaded oscillation. The design responses are unglamorous and effective — attach the context the receiver actually needs, specify acceptance criteria so rejection is cheap and early, and throttle upstream production to downstream absorption. That last one is culturally hard: slowing an automated step feels like waste, but a queue growing faster than its consumer is not throughput, it is inventory with an expiration date.

Own exceptions and recovery

The fifth discipline prices the cases that fall outside the automated envelope. A faster happy path almost always raises the absolute number of exceptions reaching humans — volume grows, and the automation's edges are where the odd cases concentrate. The coordination cost is specific: exceptions arrive stripped of context, bounce between teams hunting an owner, and consume the scarcest people, because only the experienced can handle what the machine could not.

For the topology review, two numbers matter most, and both need honest bookkeeping. First, the specialist hours consumed per exception — including the routing tour and the context-reconstruction time before anyone could act, not merely the resolution itself. Second, the ratio of that consumption to the hours the automation saves on routine cases. The ratio is the automation's real margin, and it degrades quietly as volume grows. The rule: constrain scale when recovery demand approaches the capacity the happy path releases. A deeper treatment of exception-path design is its own discipline; here the point is narrower — the exception path is part of the coordination topology, and its cost belongs in the same ledger.

One caution inherited from every queue discipline: measure absorption at the receiver's real capacity, not their nominal one. A downstream team 'absorbing' the accelerated flow by triaging faster and reading shallower has not absorbed anything; it has converted a visible queue into an invisible quality debt. Sampling downstream decision quality after an upstream acceleration is the check that catches this, and it belongs in the pilot design, not the post-incident review.

Measure the whole outcome

The sixth discipline moves the success metric from the task to the path. Value shows up — or does not — in end-to-end cycle time, total touches, outcome quality, capacity actually released for other work, coverage expanded, decision speed improved, or risk reduced. Tool usage, task duration, and adoption are supporting indicators; they measure activity at a node while the claim being tested is about the path.

The method is a disciplined before-and-after on the traced baseline: the same case types, the same path measures, output quality sampled by people who actually know the work well enough to judge it. The report-drafting story from the opening is the pattern to expect when the boundary and handoff work was skipped — task time collapses, path time holds, and the difference is the tax. The rule completes the loop: scale when the path improves and the remaining coordination is visible and owned; redesign when the path is flat; and treat a flat path with a celebrated task metric as the diagnostic it is.

The research context is worth holding at the right distance. Coordination analysis of the kind Ju formalizes asks when the work between activities is itself removable, as distinct from the activities being automatable [1]. That is a hypothesis about categories of work in general. A topology review is the empirical instrument for one workflow in particular — and every workflow I have traced has surprised its own managers somewhere, which is the practical argument for measuring over modeling when the decision is local.

Reading the map

The six disciplines run as one review, and its output fits on a wall: the traced path, links colored by bin, boundary work quantified, handoff contracts marked, exception costs attached, and the path-level baseline beside the proposal's claims. Decision-makers can then do what a task-level case never lets them do — see what the automation actually buys, at what coordination price, and where design work would change the answer.

EXHIBIT 2

The scale decision each discipline informs

Map work between tasks

Automate only after leaders can see whether the proposed change removes or relocates coordination.

Distinguish productive coordination

Keep valuable coordination, redesign avoidable routing, and automate only the stable remainder.

Find boundary work

Include boundary work in the value case and redesign the interface or process when it dominates.

Design handoffs deliberately

Throttle or reshape upstream automation when the receiving system cannot absorb the volume.

Own exceptions and recovery

Constrain scale when recovery demand exceeds the capacity saved on ordinary cases.

Measure the whole outcome

Scale when the workflow result improves and the remaining coordination is both visible and owned.

The wall format is not decoration; it is how cross-functional truth gets agreed. Stakeholders who would dispute a spreadsheet accept a traced case they can follow with a finger.

The map's second use is sequencing. Most workflows contain several automation candidates, and the topology reveals which order multiplies value: automating a step whose downstream queue is already drowning buys inventory; fixing the boundary translation first buys margin for everything after. Reviews that produce a 'not yet, and here is what first' verdict are the map earning its keep — the alternative was learning the sequence from the failure.

The costliest single error the map prevents: declaring capacity released before the path confirms it. Business cases claim the analyst hours the task no longer needs; budgets absorb the claim; and the hours, in reality, went to

boundary and exception work that nobody counted. The function is then staffed for the fictional savings and runs hot for a year. Counting released capacity at the path level, after the topology settles, is the discipline that keeps automation programs solvent.

Making the map a habit

The review installs lightly:

- Map one high-volume workflow using observed cases rather than process documentation alone.
- Label every coordination point as valuable, necessary but reducible, avoidable, or created by the intervention.
- Pilot changes at the boundaries and exception path as well as the automated task.
- Review end-to-end measures and specialist load before expanding volume or authority.

The review also needs a sponsor prepared for what it finds, because topology findings implicate design choices with authors. The approval step that traces to zero catches was somebody's governance win; the shadow reconciliation belongs to a team that built it in self-defense. Presenting findings as system behavior rather than individual fault — the same discipline incident reviews learned decades ago — is what keeps the second review welcome. A map that becomes a blame instrument gets no more honest cases to trace.

The first review takes about two weeks of part-time effort for a substantial workflow — a dozen traced cases, a few observation sessions, one wall. Run it on a workflow with an automation already proposed, so the findings land in a live decision rather than a methodology showcase. The second review is faster; by the fourth, tracing is a habit the analysts apply without being asked, which is the real installation.

Ownership belongs to the workflow owner, with the automation or AI program function supplying method and cross-workflow pattern reading. The pattern reading matters: boundary work clustering around the same downstream system across reviews is an integration investment case; exception costs clustering around the same case class is an upstream data or policy fix. Individual reviews find taxes; the portfolio view finds the tax code.

What the record should show

A topology practice leaves evidence:

- Traced paths with touch, queue, and rework baselines — before and after each intervention.
- Coordination links binned by value, with the created-by-intervention bin priced.
- Boundary-work minutes per case, by category and by the seniority performing them.
- Handoff queues within design tolerances, and throttles that operated when they were not.
- Exception hours against happy-path savings, tracked as volume grows.
- Path-level outcomes — cycle time, quality, released capacity — confirming or refuting each scale decision.

Timing the measurement window honestly matters as much as choosing the measures. The weeks immediately after an automation lands are the least representative of its life: novelty effort masquerades as boundary work, and learning-curve friction masquerades as permanent cost. Measure the path once the topology settles — usually one

to two quarters in — and then keep the baseline alive, because coordination regrows around any tool as the workflow's cases and people change. A topology mapped once is a photograph; the practice is closer to a garden.

The portfolio-level number worth an executive's attention is the sum of created-by-intervention coordination across all automations, trended quarterly. Flat or falling while automation grows means the design discipline is working. Rising faster than automation value means the program is buying task speed with path capacity — a trade that ends with the coordination tax consuming the program.

The strongest counterargument

Coordination is not inherently waste. Complex work needs consultation and cross-functional judgment. The tax is coordination whose purpose is unclear, whose route is avoidable, or whose cost was shifted out of view. Redesign should preserve necessary collaboration.

Nor is the map an argument against ambitious automation. Its verdicts regularly run the other direction: a workflow whose coordination is mostly avoidable routing and re-keying is a candidate for deeper automation than anyone proposed, because the topology shows the interaction structure would simplify rather than strain. The map is neutral machinery. What it removes is not ambition but blindness, in both directions.

The objection from speed — that mapping topologies slows an automation program that competitors are running faster — undercounts what the map prevents. The program that skips the review ships more automations per quarter and accumulates unpriced coordination debt in every one; the debt compounds into the flat outcome curve and the burned-out specialist layer that stall such programs in year two. Two weeks of tracing per workflow is the carrying cost of knowing what the automation actually does. It is the fastest sustainable speed available.

The shift

The shift is from automating tasks to redesigning paths: the task is a node, the value lives on the edges, and the topology — mapped from observed cases, binned by what each link produces, priced at the seniority that carries it, and owned by someone who will keep the baseline alive — is what separates automation that compounds from automation that merely relocates work into the dark.

Leaders do not need to run the reviews themselves; they need to change what counts as done. An automation is complete when the path improves, the remaining coordination has owners, and the released capacity survives a path-level audit. Holding that definition — against vendors, against internal enthusiasm, against their own task-metric dashboards — is the executive contribution, and it is sufficient.

The report-drafting team did eventually get its fourteen hours — eight months later, after an integration killed the re-keying, a provenance banner killed most of the explaining, and the added approval step was retired on evidence of its own uselessness. The tool never changed. The topology did.

Notes and sources

[1] Harang Ju, "When Coordination Is Avoidable: A Monotonicity Analysis of Organizational Tasks," 2026 preprint. https://arxiv.org/abs/2602.18673

[2] NIST, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," July 2024. https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.600-1.pdf

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