There Is No End-to-End Plan. There Can Be an End-to-End Decision.

Autonomy Platform · Public Blog

PERSPECTIVE

Strategy August 2026

There Is No End-to-End Plan. There Can Be an End-to-End Decision.

Lora Cecere is right that end-to-end supply chain planning is a myth: the only connector between tactical planning and transportation was always order management, and thirty-five years later it still is. The fix is not another integration. It is to stop connecting applications and start connecting decisions, with one vocabulary, one context layer, and a cascade that passes context, guardrails and targets rather than plans.

In August 2026 Lora Cecere published The Myth of End-to-End Supply Chain Planning, a piece that opens with a story from 1993. She was implementing demand and supply planning at Manugistics; her colleague Ellen was implementing logistics planning. Their goal was end-to-end planning for a client. They failed. When they asked the wider organisation for help, they were told that the only possible connection between tactical supply planning and transportation planning was order management. So they mapped TMS to order management, and order management to forecasting and distribution planning. On the same day, the company issued a press release about end-to-end planning. They laughed.

Her verdict on the intervening decades is one sentence: *"Thirty-five years later, not much has changed."*

I have spent those same decades in this industry, a good stretch of it at Kinaxis, and I am not going to argue with her. She is right. I would go further: the phrase "end-to-end" now appears on essentially every vendor site in this market, including on sites belonging to companies whose own products cannot pass a planned order from tactical supply planning to procurement without a round trip through an ERP. If you want a single test that separates the claim from the reality, ask a vendor to draw the path a *decision* takes from a demand signal to a purchase commitment to a tendered load, and to name what travels along each arrow. Most of the arrows will turn out to be a transaction, an overnight batch, or a person.

So this post is not a rebuttal. It is an attempt to take her argument as a specification and say what we have built against it, what that requires you to change architecturally, and, at the end, what we have not solved.

The myth has a specific mechanism

The reason end-to-end planning does not exist is not that vendors are lazy or that the math is missing. It is structural, and there are three parts to it.

The connector is a transaction, not a decision. Order-to-cash and procure-to-pay are the load-bearing integrations in every supply chain stack on earth. They carry quantities, dates, and identifiers. What they cannot carry is *why*: the reasoning that produced the quantity, the bounds the decider was working inside, what they expected to happen, and how confident they were. Roger M. put this well in the comments on Lora's

LinkedIn post: “Most stacks are built for transactions, not streams, so the insight dies at the handoff unless someone manually carries it forward. That manual handoff is the connector the stack never built.”

The vocabulary is different at every hop. Lora’s line is exact: *“Each system has a different definition of location, item, events, order, and purchase orders.”* A site in the TMS is a dock. A site in the planning system is a stocking point. A site in the ERP is a plant with a storage location under it. You cannot compose decisions across systems that disagree about what a site is, and no amount of middleware turns three definitions into one.

The feedback that would tell you whether any of it worked arrives quarterly and aggregated. A cross-functional handoff that goes wrong shows up as a service miss or an expedite cost three months later, confounded with everything else that moved in the period. Nobody can improve a connection they cannot measure, which is why these seams have survived thirty-five years of investment.

Her Figure 1 is the most useful thing in the piece

She introduces it as “my crude drawing from lunch with a client.” It is not crude, and I think she undersells it badly, so I want to reproduce it and then read it back to her.

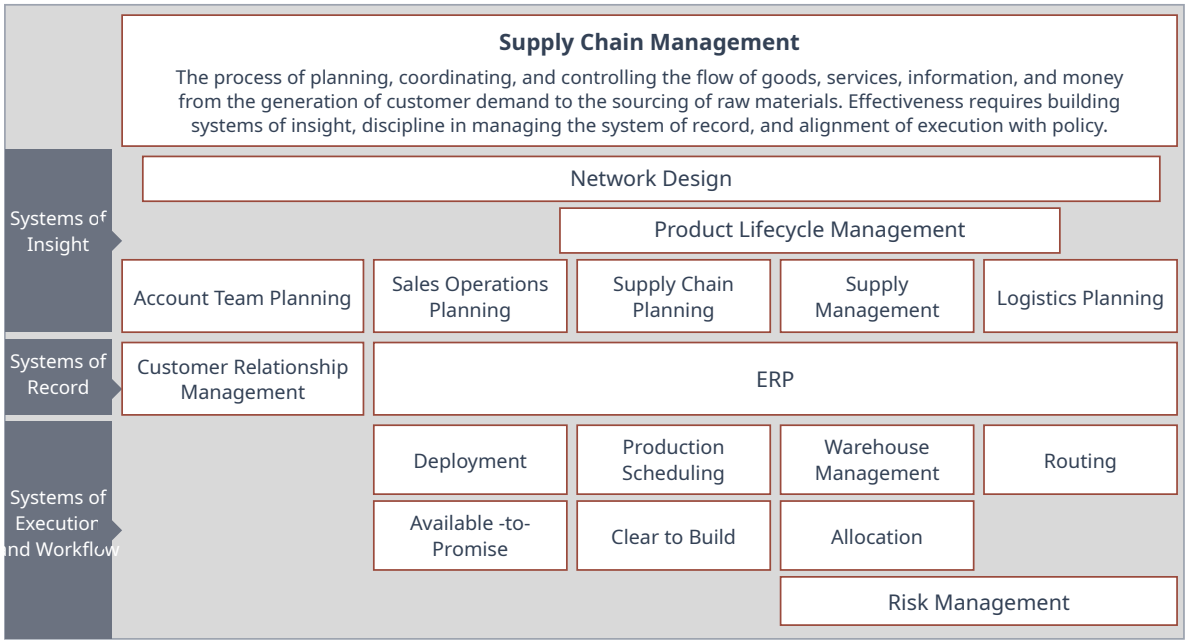


Figure 1 from Lora Cecere, The Myth of End-to-End Planning (August 2026), redrawn faithfully and reproduced with credit. The layout, the cells and the wording are hers.

Three things about how it is drawn.

The bands are horizons and the columns are domains. Read it left to right and you get the functional split: account team, sales operations, supply chain, supply management, logistics. Read it top to bottom and you get insight, then record, then execution. That is a tier-by-domain matrix, which is exactly the shape a decision architecture has to have.

Each column already carries its execution counterpart directly beneath it. Sales Operations Planning sits above Deployment and Available-to-Promise. Supply Chain Planning sits above Production Scheduling and Clear to Build. Supply Management sits above Warehouse Management and Allocation. Logistics Planning sits above Routing. The vertical pairing is right, and it is the pairing that matters, because the planning decision and the execution decision in the same domain are the two ends of one cascade.

And then there is a band wedged between them. The only box in the entire diagram that spans the full width is ERP. The insight layer and the execution layer *of the same column* can reach each other only by going down through a transactional system and back up. That is the 1993 Manugistics story rendered as a picture, and it is why she can say the connector is still order management thirty-five years later.

Two further details are worth naming because they are not accidents of a lunchtime sketch. **Network Design spans the entire width at the top**, which is correct: the shape of the network is the one decision that constrains every column beneath it. And **Risk Management is a box in the bottom right, under execution, with nothing connecting it upward**. That is not a drafting shortcut. That is the market she is describing, drawn accurately: risk detects, and has no path to the decisions it ought to change.

The separation is the error

Cell by cell, her map is our product:

Her cell	What it is in Autonomy
Network Design	**not addressed.** We optimise the parameters of a network you already have, not its structure. See below
Product Lifecycle Management	the Portfolio and Lifecycle decision domain
Sales Operations Planning	Demand Shaping, plus S&OP reconciliation at the Strategic tier
Supply Chain Planning	Supply Planning: demand-supply balancing, multi-echelon inventory, capacity feasibility
Supply Management	Procurement, supplier reliability, freight procurement
Logistics Planning	the Transport domain: lane capacity and service commitments
Deployment	replenishment
Available-to-Promise	allocated available-to-promise
Clear to Build	pre-build commit and build-ahead authorization
Allocation	allocation, at the tactical tier and again at execution
Production Scheduling	the Production Scheduling domain: capacity and build authorization
Warehouse Management	the Warehouse domain: dock scheduling and load planning
Routing	carrier selection, load planning, tender management
Risk Management	the Risk Engine, but not where she has drawn it

Three of those are not really a mapping. Production Scheduling, Warehouse, and Portfolio and Lifecycle are the names of our own decision-domain pages. We arrived at her vocabulary independently, which is either reassuring or unsurprising depending on how much you believe this industry has genuinely been confused about what the functions *are*. It has not been. The content of her map is right.

The **topology** is where I part company with her, and it is not a small disagreement. It is the whole thing.

Her three bands are not a neutral taxonomy. They are three populations of software, stacked, and the middle one is load-bearing: Systems of Record is *how the top talks to the bottom*. Insight sits above, execution sits below, and a decision gets from one to the other by being written into a transaction.

That separation is the error. Not the drawing of it, which is accurate reportage of how the market is built, but the thing being drawn.

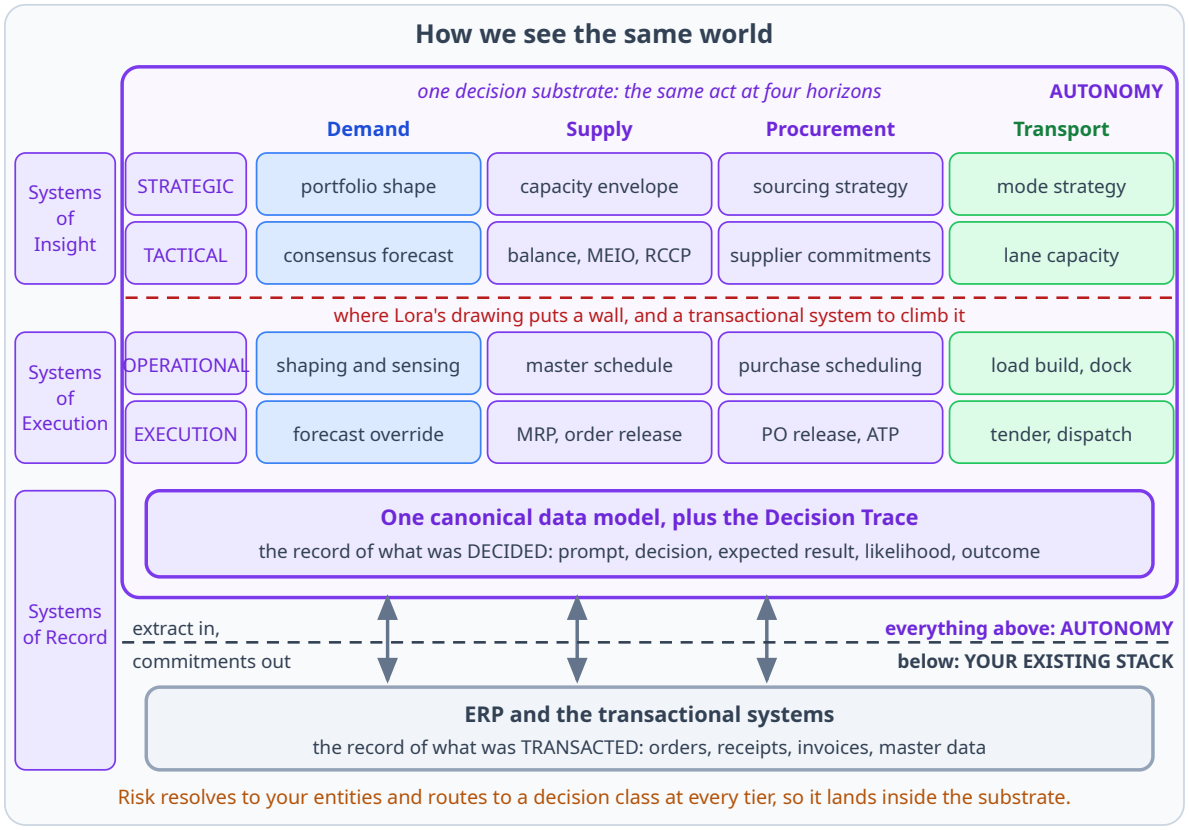
Insight and execution are not two kinds of system. They are the same act at two horizons. An agent that decides how much to buy is the thing that buys. An agent that decides which carrier gets the load is the thing that tenders it. There is no moment where a conclusion is handed across a membrane to a different population of software that carries it out, because the deciding and the doing are one event with one author. What differs between her top band and her bottom band is not the *kind* of work. It is the horizon: months and weeks above, hours and minutes below. A horizon is a gradient, not a wall.

Once you stop separating them, the record has nowhere to be except underneath. It stops being the conduit between two layers and becomes the ground both stand on.

ERP does not disappear when you do that, and it is worth being explicit about where it goes, because “we replaced your ERP” is a claim we are not making. It moves to the bottom, which is where it was always most useful, and it keeps every transaction it holds today.

That leaves two systems of record rather than one, and the distinction between them is the cleanest way I know to say what we actually add. **The ERP is the record of what was transacted**: orders, receipts, invoices, master data. **The canonical model and the Decision Trace are the record of what was decided**: what prompted it, what was chosen, what result was expected, how likely that result was, and what actually happened. Those are different facts about the same business, and no ERP schema has ever had a column for the second set, because it was never the ERP’s job to hold them.

So the line between us and your existing stack is not fuzzy, and it should not be drawn as though it were. Everything above that boundary is Autonomy. Everything below it is what you already run. What crosses it is extraction on the way in and commitments on the way out. What changes is not that the ERP matters less. What changes is that it is no longer *in the decision path*.



Same content, different topology. Insight and execution are one substrate at four horizons, over two systems of record: ours holds what was decided, yours holds what was transacted.

Three things follow from moving that band, and each one answers a complaint she makes in her own prose.

The why survives, because nothing has to cross. Her first mechanism is that the connector carries quantities and dates but not the reasoning. That is unavoidable when a decision must become a transaction to reach the thing that acts: a purchase order has fields for quantity and date and no field for “the calibrated probability this covers the demand it was sized against.” When the deciding and the doing are the same event, there is no serialisation step to lose it in. What gets written is the decision itself, carrying what prompted it, what it expected, and how confident it was.

The record can be written by the decision rather than reconstructed from its wake. This is what makes the four Inspect fields possible at all. You cannot recover an expected result and a calibrated likelihood by reading a transaction afterwards, because they were never in it. You can only have them if the thing that writes the record is the thing that made the call.

Feedback stops having to climb. Her third mechanism is that the signal telling you whether any of it worked arrives quarterly and aggregated. It arrives that way because it has to come back up through the same lossy channel it went down, by which point it is a KPI rather than an outcome. With a shared record underneath, the outcome attaches to the decision row that caused it, at the grain that decision was made at.

And the Risk Management box moves for the same reason. It is not a box any more. Signals resolve to entities, route to named decision classes, and those classes carry their tier, so risk arrives at Strategic as a sourcing question, at Tactical as an allocation question and at Execution as a tender question. That is the content she drew stranded in the corner, given somewhere to go. It could not have gone anywhere in her topology, because the only route upward was through the transactional band.

Two of her cells are honestly not us, and one of them is the box at the top.

Account Team Planning and Customer Relationship Management are the commercial column, and we do not touch it.

Network Design we have not addressed at all, and it is worth being exact about the line, because it is a line a lot of vendors blur. Network design decides the *structure* of a supply chain: where to put a plant or a

distribution centre, which to open, which to close, what the footprint should be. We do none of that. What we do is **optimal parameterisation of the network you already have**: how much capacity to authorise at the sites that exist, how to allocate volume across the suppliers already qualified, which mode to use on the lanes already contracted, where to hold buffer across the echelons already in place. That is a real and valuable problem, and it is the one every planning decision below it actually depends on. It is not the same problem as designing the network, and the topology is a given for us rather than a variable.

So her top box stays her top box. Ours would be a different box sitting just beneath it, and drawing them as the same thing would be the kind of claim this post exists to argue against.

So when she says the industry should be using AI frameworks to work out what decisions should be made and who should make them, my reaction is that she had already drawn the answer to the first half. The four questions are what you have to answer to make her own map executable, once you have stopped stacking it in three layers.

She did not just diagnose it. She wrote the specification.

The part of her piece I think most readers will skim past is the four questions. She says we should be using what AI frameworks now make possible to define:

> *"What decisions should be made? Who should make the decision? At what frequency should the decision be made? What defines a good decision?"*

That is not a rhetorical flourish. That is a requirements document, and it is a better one than most of the RFPs I have read. Each question implies a specific object that has to exist in the software, and the reason no current stack answers them is that none of those objects exist in it.

Her question	What the software actually needs
What decisions should be made?	A named, enumerable registry of decision classes, each with the grain it is made at: which product level, which geography level, which time bucket. Not screens. Decisions.
Who should make the decision?	An owner per decision class, and an *envelope* for that owner: the bounds inside which it may act without asking, declared as data rather than implied by which module the button is in.
At what frequency?	A cadence bound to the horizon of the decision, running automatically. Weekly for the capacity and sourcing envelope, daily for balancing, hourly for the site schedule, continuously for order release.
What defines a good decision?	An objective the decision is scored against, a calibrated probability that the expected result will actually be realised, and an outcome measured afterwards and attributed back to that specific decision.

Answer the four honestly and you have not built end-to-end planning. You have built something else, which is what Mark Robinson identified in the comments under her post, and it is the single sharpest sentence in that whole thread:

> *“Perhaps we’ve been trying to connect entire applications, when we should be connecting decisions.”*

Yes. That is the whole thing. Paul Malott arrived at the same place from a different angle in the same thread: the opportunity is “designing end-to-end decision architecture” rather than end-to-end system integration. Applications will never merge, because they belong to functions that will never merge. Decisions can compose, because a decision has a shape: a scope, an owner, a horizon, an objective, and an outcome.

What we built at Azirella, and what it costs you architecturally

Autonomy is our attempt at that decision architecture. Four things carry it, and each of them is a direct answer to something in Lora’s piece.

1. One vocabulary, not three

Lora’s closing instruction is to “build a unified data model across the solution stack and define the canonical, rules-based ontological frameworks and a planning master data layer.” We took that as non-negotiable before writing an agent. Every entity in Autonomy is the AWS Supply Chain Data Model: one definition of site, product, transportation lane, trading partner, geography, purchase order, transfer order. Demand, supply and transport agents read the same rows in the same database. There is no translation layer between the domains because there is nothing to translate.

The cost of that decision is real and I want to be plain about it. It means the connectors are thick and the ERP mapping work is the hardest part of an implementation, because we insist on landing customer data in canonical form rather than letting each domain keep its native shape. We

also keep the source system's original value alongside our derived one rather than overwriting it, so a planner can always see what their ERP said and what we suggest instead. That comparison turns out to be one of the more valuable things we produce in the first month of a deployment.

2. A Context Engine: her market translation, made into an object

Lora has been arguing for outside-in planning for well over a decade, and her definition of it is the most compact statement of what a context layer is actually for:

> *"Modeling based on channel and supply network signals. The translation of market signals from the customer's customer to the supplier's supplier with bi-directional flows."*

Read that carefully, because there are two jobs in it and the industry has only ever attempted the first. The first is **sensing**: get the market signal into the building. The second is **translation**: carry it from the customer's customer to the supplier's supplier, in both directions, so it lands as something each function can act on. Sensing is a data problem and it is largely solved. Translation is a decision problem and it is barely started.

Her paragraph on risk technology in that post is what happens when you do the first and skip the second: "Risk management technologies search for anomalies and provide early alerting, but there is no common data model, consistent definition of events, or workflows." Exactly. The alerting market solved detection and stopped. An alert with no defined consumer is a notification, and a notification is work you have created for a human, not work you have removed.

It is not a sweep of the internet. It is a sweep of *your* supply chain.

This is the part that most distinguishes a context layer from a news feed, and it is worth being blunt about. The Context Engine does not go and read the world. It reads the specific slice of the world that touches this customer's entities: their sites, their lanes, their products, their suppliers, their carriers, the geographies they actually operate in, the industry they are in, and the market their products actually compete in. Everything else is noise with good production values.

The mechanism is that every signal in the catalog declares what natural key its source carries, and which canonical entity that key resolves to. A storm warning arrives as a latitude, longitude and radius, and resolves to the sites and lanes inside it. A product recall arrives as a GTIN or an NDC, and resolves to products through the customer's own identifier registry. A carrier safety downgrade arrives as a DOT number and resolves to a trading partner. A lane rate move arrives as an origin-destination pair and resolves to a lane. A national CPI print is not "macro and therefore unscoped": it is a country-level geography that resolves to every site in that country, which is coarse but still discriminating for a customer operating in six of them.

Two consequences follow, and both matter more than they sound.

A signal that resolves to none of the customer's entities is not stored. It raises. The reason is not tidiness. Once a row is written with no scope, "matched nothing" and "applies to everything" become the same row, and a signal that quietly means "applies to everything" is exactly how a context layer starts generating noise it can never be cleaned of. In the current catalog there is no signal without a declared scope. The unscoped escape hatch exists for a source that genuinely carries no key, and nothing uses it.

Relevance is structural, not statistical. A signal is not scored for interestingness by a model that might be wrong. It either resolves to entities the customer owns or it does not. That is the difference between a curated context layer and a supply chain news alert with a relevance percentage on it.

Every signal is declared on three axes

Beyond scope, each signal type declares two more things: which decision classes it drives in each domain, and how it behaves as a stochastic process when the digital twin needs to simulate it. Where it lands, what it changes, how it moves.

The middle axis is what produces Lora's translation. One severe weather event, declared once, fans out three ways: to inventory buffering in supply, to demand sensing in demand, to equipment repositioning in transport. A consumer price index move routes to forecast adjustment on one side and freight procurement on the other. The routing is data, validated against the decision registry, so a signal cannot be added to the system without saying what it is supposed to change.

And because every decision class in that registry carries its planning tier as well as its domain, naming the decisions a signal drives also names the tiers it belongs to. The same raw feed is therefore curated differently at each level rather than broadcast identically to all of them. A sustained freight capacity index shift is a driver for the strategic tier's mode and contracted-capacity decisions at category, region and month grain. The same feed, this week, is a driver for tactical lane commitments at family, state and week. The same feed, this morning, is a driver for a tender

decision at product, site and hour. One signal, four different pieces of information, because a driver only means something at the grain the decision is made at.

That is what stops the insight dying at the handoff. The signal does not land in a risk dashboard for someone to carry forward manually. It lands on the decisions it is supposed to move, plural, in every domain and at every tier that has one.

And nobody has to read it

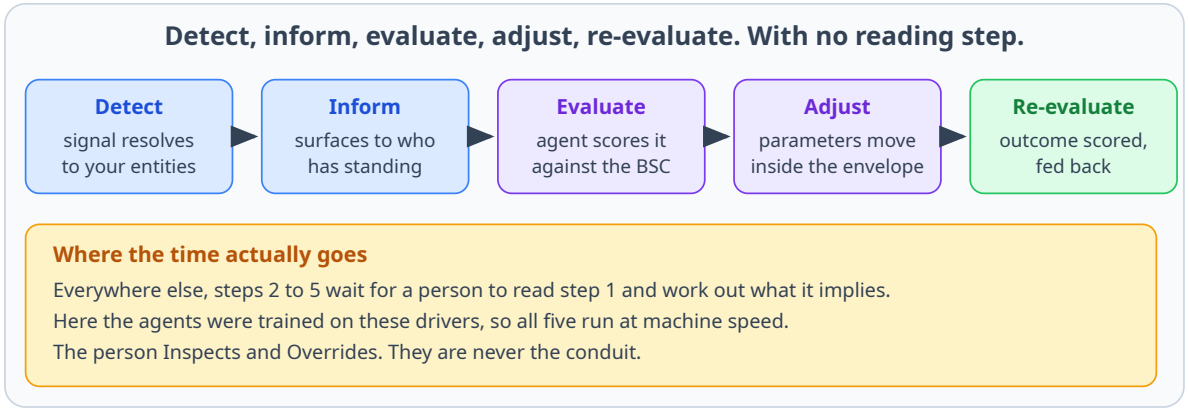
This is the part I would most want a buyer to press us on, because it is where the difference is largest and where it is easiest to sound the same as everybody else.

Every risk, visibility and control-tower product in this market terminates at a person. It detects, it informs, and there it stops, because the next move requires someone to read the alert, work out what it implies for their part of the plan, decide what to change, make the change, and later find out whether it helped. Five steps. The product does one of them.

The other four run at human speed, and they run at human speed *in series*: the planner reads it Tuesday, works out the implication Wednesday, changes the buy on Thursday, and the question of whether it worked is asked at the month-end review or not at all. Meanwhile the disruption has been compounding since Monday. That is the actual cost of the missing connector, and it is not a cost you can fix with a better dashboard, because the dashboard is not the slow part. The reading is.

In Autonomy the agents have been trained on these drivers. Not shown them at runtime and asked to cope: **trained on them**, as feature stressors in the digital twin, so responding to a lead-time shock or a capacity index

shift or a recall is a learned reflex rather than an interpretation someone performs. That changes what the five steps cost:

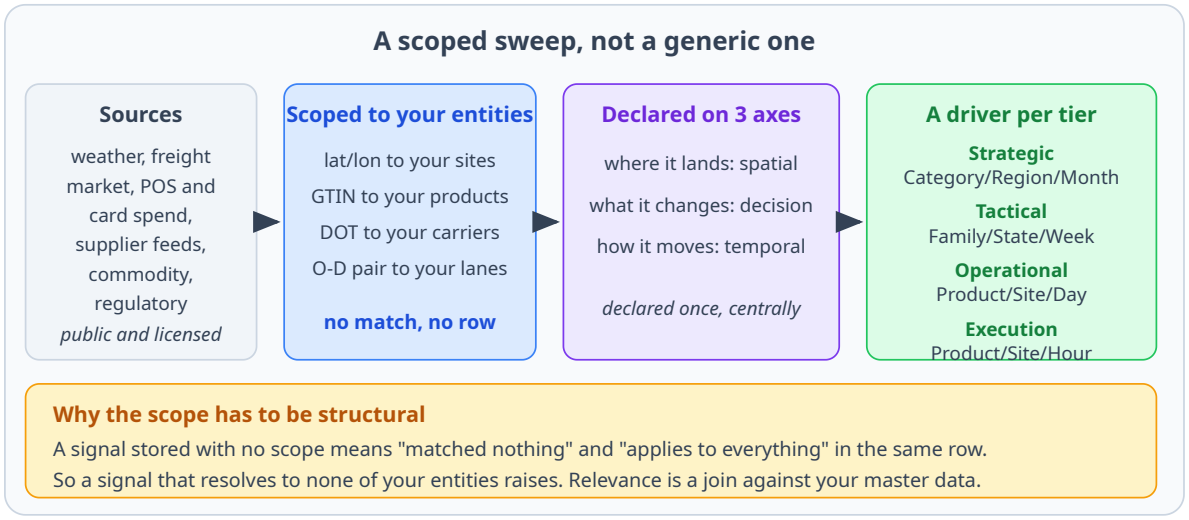


The dashboard was never the slow part. The reading was.

Two things that follow, and I want to be careful with both.

The human is not removed, they are moved. Inform still fires, and Inspect and Override are still there and still matter. What changes is that they are no longer *load-bearing*: the loop closes whether or not anyone opens the row, and the person intervenes because they know something the agent did not, rather than because the machine is waiting on them. That is the inversion in AI·IO·ML, applied to context rather than to plans.

And this is the whole argument for latency. A disruption's cost is a function of how long you take to respond to it, which means every hour spent in the reading step is paid for in inventory, expedite freight or missed service. The industry has spent a decade shortening step one, from weekly reports to real-time feeds, while leaving steps two through five exactly where they were. Detecting faster and responding at the same speed does not compound. It just means you spend longer knowing.



The same feed becomes four different drivers, because a driver only means something at the grain the decision is made at.

But the Context Engine only does half of her translation, and I want to be precise about which half. It gets the signal to the right decisions, at the right grain, at the moment it arrives. Carrying the consequence *across echelons*, from the shelf back to the supplier's supplier and forward again, is not an ingestion feature. That is the cascade, and it is the next section.

3. A Risk Engine with a learned layer under the threshold layer

Every planning system has threshold detectors: projected stockout, overstock, lead time variance. We have those too, and we have generalised them so a stockout detector works on any entity with a stock and a demand stream rather than being written three times for three domains.

Underneath sits the part that is harder to copy. We run graph neural networks over the supply network itself, and their risk heads are learned rather than computed by rule: how critical a node is as a single point of failure, where the structural bottleneck is, how concentrated supply is on

one partner, how resilient a subgraph is. These are differentiable, which means they improve against realised outcomes. Classical graph algorithms cannot do that. Betweenness centrality is the same number today as it was last year no matter how many disruptions taught you something.

The structural risk scores publish back into the same alert surface as the threshold detectors, so an operator sees one list. But the two layers answer different questions: the threshold layer says “this SKU at this site will run out on Thursday”; the learned layer says “this supplier is where your network actually breaks, and it has been for six months.”

4. A cascade that passes context, guardrails and targets, never a plan

This is the core of it. It is the direct answer to the Manugistics story, and it is the other half of market translation.

Lora’s complaint is that there is no process flow to map planned orders from tactical supply planning to procurement or to transportation. Our position is that mapping planned orders is the wrong ambition. A planned order is an *answer*, computed at one grain against one objective, and shipping it downstream as a directive means every tier below is executing a decision that was made without the information that only exists further down. That is why the aggregate plan is always infeasible when it hits the site: the constraint that binds at the line is invisible one level up, where it has been averaged away.

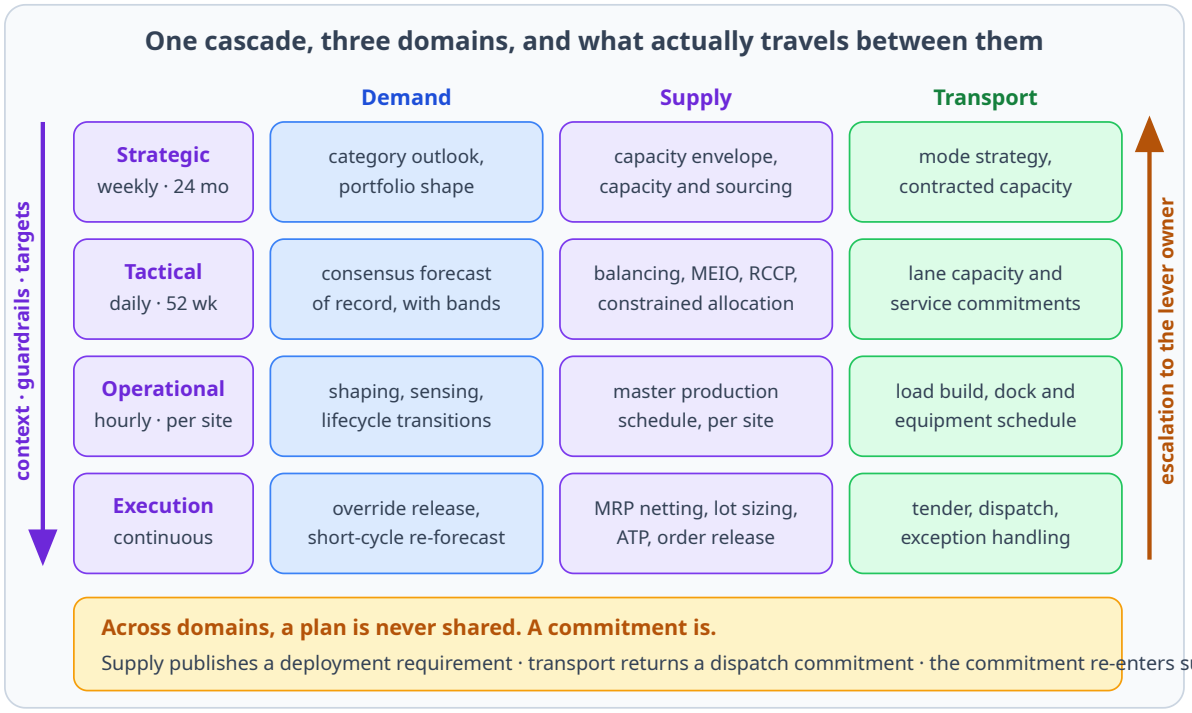
So down the cascade we send three things, and never a plan:

- **Context**: what the tier above is seeing and assuming, including the exogenous signals it routed.

- **Guardrails**: the bounds inside which the tier below may act, declared formally as obligations, permissions and prohibitions, each with a named escalation target if it is breached.
- **Targets**: what good looks like at this tier, expressed on a balanced scorecard rather than a single cost number.

Each tier then *generates its own plan* at its own grain against its own scorecard, inside the envelope it inherited. Strategic sets the capacity and sourcing envelope monthly-grained over two years, across the network as it stands. Tactical balances demand and supply and allocates constrained volume weekly. Operational builds the site's master production schedule daily. Execution nets requirements and releases orders continuously through the day. Same twin, same physics, four grains.

And critically, the path runs upward too. When a tier hits a constraint it cannot relieve inside its own envelope, that escalates one hop at a time to the tier that owns the lever. A plant pinned against its capacity ceiling for six straight weeks is not an execution failure, and no amount of clever sequencing at the site will fix it. It is a signal that belongs to whoever can authorise a shift, a second source, or capital. The escalation is not a breach and it does not stop the agent working; it informs the tier that can act, on the next cycle.



Four tiers, three domains, one twin. Down the cascade: context, guardrails, targets. Up it: the constraint the tier below cannot relieve.

Across domains: contracts, not plan rows

The same principle governs the horizontal seam that defeated Lora and Ellen in 1993. Supply does not write transport's plan and transport does not write supply's. Supply publishes a *deployment requirement*: this quantity, from here to here, needed by then, with this much date flexibility. Transport answers with a *dispatch commitment*, or declines and says why. The commitment then re-enters supply planning as a constraint on the next cycle, and the realised outcome of that commitment, whether the lane actually delivered, is fed back as evidence rather than as a complaint.

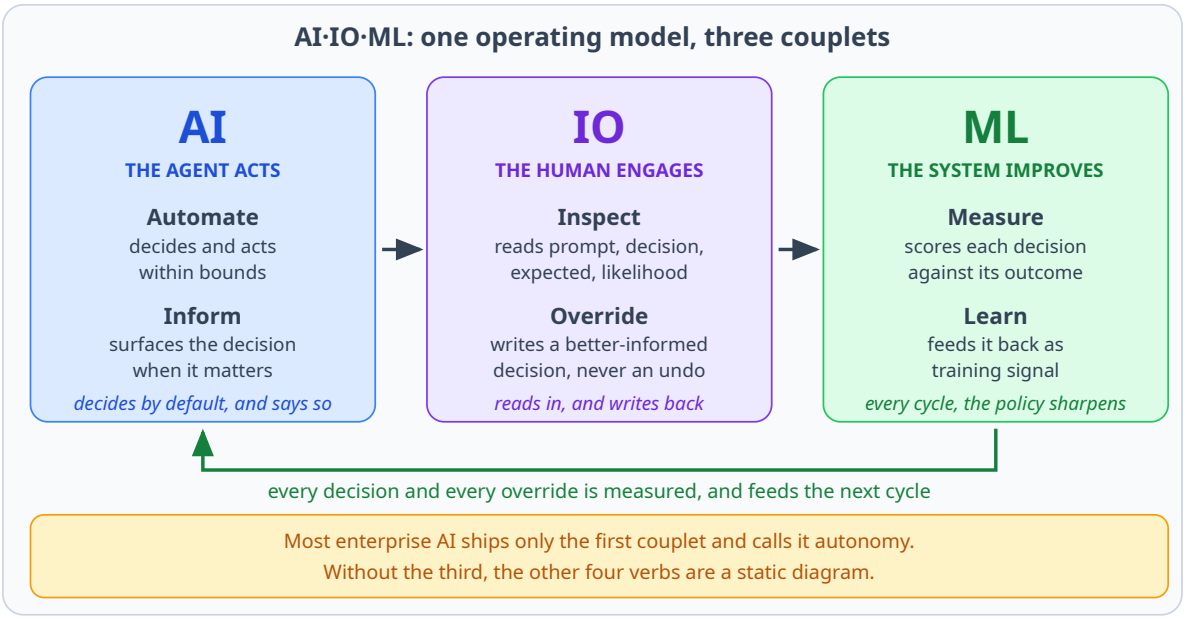
That is a narrow, versioned, auditable interface between two domains that keep their own objectives. It is deliberately not an integration and deliberately not a merge. Charmaine Huang made this point in the

comments too: without the authority to impose across functions, what you actually have is orchestration, and orchestration needs a contract, not a shared spreadsheet.

It is also, I think, the literal mechanism behind the last three words of Lora's outside-in definition. "Bi-directional flows" cannot mean a data feed running in two directions, because data in the reverse direction is just a report. It has to mean that a downstream constraint changes an upstream decision. A commitment declined by transport, with a reason, is exactly that: it re-enters supply planning as a bound on the next cycle rather than as an exception someone reconciles by hand. Translation from the customer's customer to the supplier's supplier is not one long pipe. It is a chain of decisions, each one inheriting a bound from its neighbour and returning a commitment.

The loop: AI·IO·ML

None of the above is worth much if it does not improve. The operating model we built the product around is AI·IO·ML: one loop, three couplets. The agent acts. The human engages. The system improves. It is our answer to Lora's fourth question, the one about what defines a good decision.



The third couplet is the one that is usually missing, and it is the one that answers Lora's fourth question.

AI: the agent acts.

Automate. The agent decides, inside the envelope it was given. There is no approval queue as a structural feature. A customer can configure any decision class to require a human every time, and some should be, but that is a setting they chose, not a bottleneck we built in.

Inform. The decision has already been made. It surfaces to the people who need to know it, when it crosses a policy boundary or when calibrated confidence is low and urgency is high.

IO: the human engages.

Inspect. Any decision can be opened, and it returns four fields: what prompted it, what was decided, what result was expected, and the calibrated likelihood that the expected result is realised. That last number comes from a conformal prediction layer with a coverage guarantee, not

from a model's self-reported confidence. If we do not have a calibrated likelihood for a decision, it is routed for human inspection rather than being given a number we cannot defend.

Override. A person supersedes the decision because they know something the agent did not. Override is not undo. It is a new, better-informed decision, and it is the most valuable data the system ever receives, because it is a labelled example of the gap between the model and the operator's knowledge.

ML: the system improves.

Measure. Because the agent acted, the outcome is real. Every decision is scored against its outcome, and against the counterfactual: what would the alternative have produced. Critically, measurement runs on every decision rather than only the contested ones. A system that measures only the decisions a human touched is learning a distorted view of its own performance, and it will be confidently wrong about exactly the decisions nobody was watching.

Learn. The agents retrain on every decision, outcome and override triple. Overrides become operating knowledge, curated rather than absorbed automatically. Calibration tightens only where the record has earned it, so decisions migrate from Inform into Automate on evidence rather than on optimism. And the Inform threshold is itself learned, trading the value of a planner's attention against the probability that their intervention catches a real mistake.

That last couplet is the one most enterprise AI leaves out, and leaving it out is what turns the first four verbs into a diagram. Note also where the two halves of Lora's fourth question land: Measure is what makes "a good

decision" an observable fact rather than an opinion, and the calibrated likelihood in Inspect is what lets you say so *before* the outcome is known rather than after.

The thing that makes this loop possible is not the learning. It is the record. Every decision is written once, at the grain it was made at, with its four Inspect fields, its parent directive, its envelope, and the identity of the agent that made it, in a tamper-evident chain. Lora's four questions are answerable for any single decision in the system, individually, on demand. That was the design constraint, and everything else followed from it.

What we have not solved

If I stopped here this would read exactly like the vendor pitch Lora is criticising, and Jon Kirkegaard's comment on her post would apply to us: market the full solution, ship the bit part. So here is the honest register.

Measure is the hard part of the loop, and it is the least finished. Attributing a realised outcome back to the specific decision that caused it requires the ERP to give you a clean actual, at the right grain, tied to the right identifier. Very often it does not. Where we cannot measure an outcome, the correct behaviour is to say so and to refuse to claim the loop closed, and that discipline is worth more than the appearance of a complete feedback cycle. Any vendor telling you their agents learn continuously should be asked exactly where the actuals come from, and what happens when the answer is that the system compared the plan against itself.

The signal catalog is declared ahead of the connectors. Every signal type in the catalog has its scope, its decision routing and its stochastic behaviour declared, and the resolver that matches a signal to a customer's entities enforces the no-match-no-row rule. Wiring each individual commercial feed through that resolver is incremental work that runs per source, and a customer's first deployment will have more of the free public sources flowing than the paid ones. The contract is the thing that was worth getting right first; the feeds connect against a fixed contract rather than each inventing its own shape.

Transport is the domain where the market's own infrastructure matters most. Real-time visibility, carrier event feeds and tender responses are a network problem before they are a planning problem, and the sensible posture for us is to integrate with the visibility layer the market has already built rather than to reproduce it. Our contribution at the transport seam is the decision contract and the capacity commitment, not another tracking product.

Philippe Lambotte's objection is the strongest one in the thread, and I do not have a software answer to it. He points out that the software industry follows its customers' organisational and process maturity, and cites Kraft Heinz recombining procurement with operations thirty years after it was already organised that way. He is right that you cannot ship an architecture into an organisation that does not want the seam removed. What software can do is narrower and still worth something: make the handoff between two functions a governed decision with a named owner, a declared envelope, and a measured outcome, rather than an email and a spreadsheet. Whether the two functions then choose to align is not our call.

And the seams multiply below the level Lora drew. Shristi Upreti-Nitsch made this point sharply: manufacturing execution is often the most isolated silo of all, and the fragmentation runs *within* each pillar as well as between them. Channel planning splits B2B from B2C. Logistics fragments across inbound, warehousing, outbound and last mile. We address the vertical seam between planning tiers and the horizontal seam between domains. We do not claim to have dissolved every seam inside them.

The test

The most useful thing to come out of Lora's post is not agreement. It is that her four questions are a test any of us can be held to, and unlike "end-to-end", they cannot be satisfied by a diagram. If you are evaluating anyone in this market, including us, ask for a single decision the system made last Tuesday and require five things about it:

1. **Name the decision class**, and the grain it was made at: which product level, which geography level, which time bucket. If the answer is "it's in the plan", there is no decision object.
2. **Name the owner and the envelope.** What was it allowed to do without asking, and where is that bound written down as data?
3. **Show what it inherited.** Which higher-tier context, guardrails and targets was it acting inside, and can you follow the link upward?
4. **Show the expected result and the calibrated likelihood**, recorded before the outcome was known. Anyone can produce a confidence number afterwards.
5. **Show the outcome, attributed back to that decision.** Not a KPI for the quarter. That decision.

A stack that can answer those five for one decision can answer them for all of them, and it is a fundamentally different object from a set of applications with connectors between them. A stack that cannot is doing what Lora described: hanging agents on old architecture like icicles on a holiday tree.

There is no end-to-end plan. There never was, and the pursuit of one has consumed thirty-five years and an enormous amount of money. But there can be an end-to-end decision: one that inherits its bounds from the tier above, respects a commitment made in another domain, carries its own calibrated confidence, and is measured afterwards against what actually happened.

That is what we are building. I would rather be judged on the five questions than on the phrase.

Trevor Miles is Founder and CEO of Azirella. Read more about the operating model at AI·IO·ML, or about how the substrate learns at How Agents Learn.