



What a Decision Factory Actually Requires

Autonomy Platform · Public Blog

PERSPECTIVE

© 2026 Azirella Ltd. All rights reserved worldwide.

[Read more at azirella.com/blog](https://azirella.com/blog)

TL;DR

DeJans and Elam's novel puts the premise in one sentence: real operations are not solved once, they are decided again and again. What that commits you to is not a better optimiser. It is four genuinely different planning jobs that have to be told apart, a clear rule about what each hands the next, and an honest answer to a question most systems never ask about the customer's own data.

KEY POINTS

- A plan pushed down from the layer above is not an instruction the layer below can obey. A monthly category-by-region number meets a real changeover on Tuesday and stops being feasible. So each layer makes its own plan, and what travels down is not the plan.
- Reconcile where the hierarchy is an identity, not where it is a hand-off. Category sales are SKU sales added up, so reconciling the forecast adds information; coherence between two different decisions is invented rather than restored.
- The return path is what most systems lack. A plant pinned at its ceiling for six weeks is not misbehaving, the fence is wrong, and it does not own the lever. Escalation goes up one layer at a time, is information rather than a fault, and relief comes next cycle, never mid-flight.
- Run head to head on real network shapes, the rule-with-dials beat both the classical optimiser and a neural policy eight times out of eight. We would not have believed it without running the contest.

Adam DeJans Jr. and John Elam’s new novel, *The Decision Factory*, with a foreword by Warren Powell, puts the premise in one sentence:

Real operations are not solved once. They are decided again and again, under uncertainty, with imperfect information and real consequences.

Every planning vendor would agree with that. The interesting part is what it commits you to, and the answer is not a better optimiser. It is four *different* planning jobs that have to be told apart, a clear rule about what each one hands the next, and an honest answer to a question most systems never ask: how wrong is this customer’s data, and do we actually know?

Powell’s framework is the cleanest language I know for that. This is it without the mathematics.

Four jobs, not one plan

Almost every planning system treats “the plan” as one artefact that gets refined as the horizon shortens. That framing is the problem. There are four genuinely different jobs, they answer different questions, they run at different speeds, and the person doing each one has a different title.

Four jobs, four questions, four time buckets

the job	the question it answers	runs / plans in
Strategic S&OP / IBP	What shape should the network be, how much buffer do we carry, what service do we promise?	runs weekly, plans in MONTHS
Tactical master planning	Supply is short. Whose orders get filled, whose slip, and where does the stock sit meanwhile?	runs daily, plans in WEEKS
Operational MPS and S&OE	What do we build at this plant this week, and when a line goes down, what moves where?	runs hourly, plans in DAYS
Execution the order desk	Cut this purchase order. Promise this date. Release this works order. This lot size.	runs every 4h, plans in HOURS

These are not four views of one plan. They are four decisions, each of which can be right or wrong on its own.

Those last two lines are separate things. S&OP **re-runs weekly** but **plans in monthly buckets**; the order desk re-runs every few hours while each decision returns in milliseconds. How often you re-plan, how coarse the answer is, and how fast one decision comes back are three different questions, and a system reporting one when you asked another will mislead you about all three.

Which matters commercially, because **a plan pushed down from the layer above is not an instruction the layer below can obey**. An S&OP number is monthly, by category, by region; a plant supervisor needs a specific product on a specific line on Tuesday. Handing down a number that was feasible in aggregate and is infeasible the moment it meets a real changeover is where most planning systems quietly lose the plot.

So each layer makes its own plan. What travels down is not the plan.

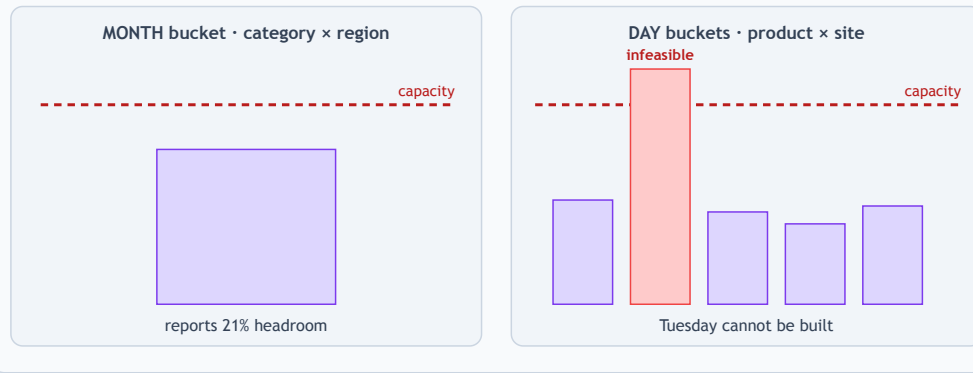
Why the detailed plan is not the aggregate plan

That sentence is the one people stop at, because the industry standard is the opposite. Set the aggregate plan first, then require each more granular plan to reconcile to it. It has a name - plan consumption - every planning system on the market ships it, and it is so established that most planners have never worked any other way. It is worth asking why we still do it.

Every level of aggregation loses detail, on all three hierarchies at once. A peak on a Tuesday gets squashed by weekly buckets. A peak in the third week gets squashed by monthly ones. A customer growing and a customer shrinking cancel each other out at the region level. None of that is a flaw in the aggregation - it is what aggregation is *for*. The question is why we then expect the detailed plan to reproduce a number that was built by discarding the detail.

And the roll-up does not merely blur the answer. On the question that matters it reverses it.

Same demand. Same capacity. Opposite conclusion.



A constraint binds only at the level it lives on. Average a bottleneck day against its neighbours and the aggregate reports headroom that nobody has.

But that is still the weaker argument. The stronger one is that they are not the same decision. A monthly volume by category and region, and a Tuesday sequence on line three, are not one quantity at two magnifications. The second depends on things the first never modelled: what the line is currently set up to run, what is halfway through, what got ordered on Friday afternoon. You cannot derive it from the monthly number at any resolution, because that information was never in the monthly number to begin with. Reconciliation assumes the two levels are deciding the same thing, and they are not.

Aggregation is not pure loss, and the honest version has to say so. It removes noise as well as signal, which is exactly why aggregate forecasts are more accurate than the sum of their parts. On one real network the variability we have to plan against spans sixty-seven-fold across individual product-location pairs and only fifteen-fold once pooled to six categories. The aggregate is a *better-estimated total* and a *worse-resolved structure*. Those are two different things, and that is precisely why neither plan is a degraded copy of the other.

So we reconcile the forecast, and we do not reconcile the plan

This is the part worth taking away, and it is not a compromise between the two positions.

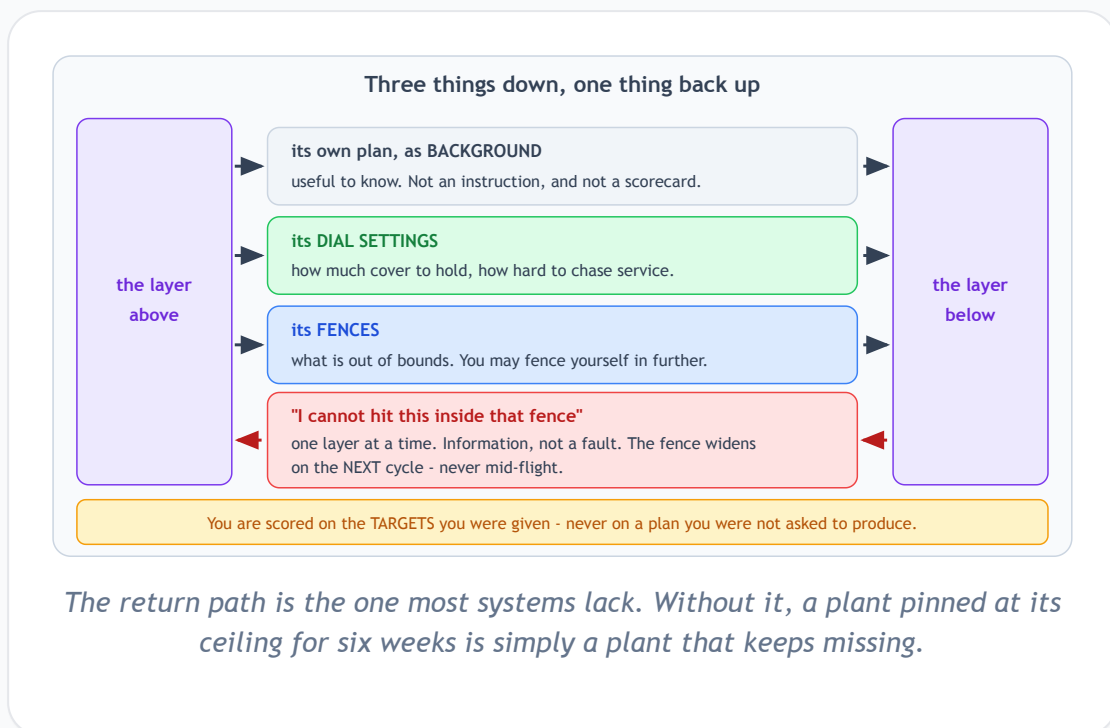
Demand is one quantity seen at several magnifications. Category sales *are* SKU sales added up. Coherence is a real property of that data, so imposing it adds information rather than inventing it - and there is a well-established statistical result showing that reconciled forecasts are no worse than unreconciled ones and usually better. Our forecasts are built that way: one demand history, aggregated to each level's own grain, so no level holds a private opinion about what the market wants.

A plan is a hand-off between different decisions. There is no identity to recover, so coherence there is not restored, it is invented - and the theorem that justifies reconciling forecasts has nothing to say about it. So the levels get targets, dials and fences instead, and each builds its own plan and is judged on its own results.

Reconcile where the hierarchy is an identity. Do not reconcile where it is a hand-off. Most planning systems apply the first rule to both.

What travels down, and what has to come back up

Three things travel down, and keeping them apart is the whole design. One thing travels back up, and most planning systems do not have it at all.



If a plant is scored against the S&OP number, the plant will hit the S&OP number and the business will lose money doing it. Score it on the targets it was given and let it find its own way there inside the fences.

A dial says “run with about three weeks of cover”; a fence says “never go below one week.” Turning a dial changes how the layer below *behaves*, moving a fence changes what it is *allowed to do at all*, and handing a dial down disguised as a narrow fence quietly removes the judgement you were paying that layer to exercise.

The andon cord: when the layer below cannot fix it

Everything above assumes each layer can hit its targets inside its fences. The case that matters most is the one where it cannot, and nothing is going wrong.

A plant pinned at its capacity ceiling for six weeks is not misbehaving - every schedule it built was legal. **The fence is the problem**, and the plant does not own the lever that moves it: adding a shift, qualifying a second supplier, opening a lane all sit one or two layers up. So it has to be able to say so, and be heard.

Three properties make that work, and we got each of them wrong first:

- **It goes up one layer at a time.** Not straight to S&OP. The layer immediately above may own the lever, and if it does not, it passes it on having added what it knows. A path that skips levels arrives without the context that would let anyone act on it.
- **It is information, not a fault.** The layer below is not refusing to work and must not be stopped, put in an exception queue, or flagged as failing. It keeps running inside the fence it has, while the message travels.
- **Relief comes on the next cycle, never mid-flight.** The layer above widens the fence at its own next run. It does not reach into a schedule that is already executing, which is the behaviour every planner recognises as the intervention that makes things worse.

That third one is the whole reason the loop is safe to automate. A supervisor that can reach down and change a decision already in flight is not a planning hierarchy; it is an escalation culture with software attached.

Four ways to decide, and which job gets which

Powell's second contribution is that there are only four honest ways to make a repeated decision under uncertainty:

Four ways to decide. Pick per job, with a reason.

A tuned rule of thumb

"When stock drops below X, order up to Y."
Fast, explainable, and X and Y are learned rather than guessed.

We use it at: the order desk

A rule with dials

The same idea, but the dials are set by experiment against a simulation of your own network, not by a consultant's default.

We use it at: master planning

Learn what a situation is worth

"Being short of this part, at this plant, in this week, costs about this much." Decide by comparing what each option leaves you in.

We use it at: the plant schedule

Play the tape forward

Roll the horizon out, solve it, do the first step, roll again. This is what a classical optimiser does every night.

We use it as: the feasibility check

Most planning arguments are really arguments about which of these four a job should use. Naming them settles a lot of them.

Two things about that picture are worth stating plainly.

The method is a choice per job, never a property of the job's name. We had a column meant to record which method each layer used; every row in it simply repeated the layer's own name. Twenty pieces of code wrote that column and not one ever read it, so it recorded a decision nobody had actually made. Each layer now carries a card saying which method it uses, why, and - the part that matters - **what evidence would make us change it**. A method with no such statement is a label, not a decision.

And the answer surprised us. Run head to head on real customer network shapes, the humble rule-with-dials beat both the classical optimiser and a neural policy trained on simulation, **eight times out of eight** - and the neural policy beat the classical optimiser only six times out of eight. The fashionable option was not reliably better than the one it was supposed to replace. The interpretable option was the best planner we had, and we would not have believed it without running the contest.

Knowing how wrong you are

A forecast of 10,000 is not a plan input. A forecast of 10,000 that might be 8,000 or might be 15,000 is a different thing entirely, and the buffer you carry is a function of that spread. So we will not accept a single number: every forecast, every lead time, every yield carries a range. Where your own history supports one, the range is **earned from it** - what we predicted before, against what actually happened - and we report, per uncertainty and on day one, which ones your data can and cannot support.

Where it cannot, we synthesise a range instead, because a business with no receipt history still has to plan. That is the honest answer and it is not the whole story: **the decision reading that range cannot currently tell which kind it got.** A range earned from four hundred real receipts and one assembled from a supplier-master constant look identical at the point where a buffer gets sized. We know it, we are fixing it, and it is the one open defect in this piece - which is a better position than the version of this paragraph I wrote last week, claiming we simply did not invent spreads at all.

That check found something probably true at your company too. The date a supplier *promises* against the date goods *actually land* is the most universally recorded pair of numbers in any ERP on earth, and across every one of our ERP-connected customers we were extracting it on **zero of 5,599 purchase orders**. One connector was reading both halves of the pair and writing neither; another was reading lead time off the wrong record entirely. Nobody had noticed, because a missing number raises no alarm - it quietly becomes a default.

We also tried the fashionable alternative - generating thousands of simulated futures and planning against them - and it measured **nine to seventeen points worse on order fill, with cost out by thirty to two hundred per cent**. The simulator earns its keep in training, where being wrong is cheap, and stays off the live planning path.

The two rules we now apply to every number

The first is that **every planning number has two halves, and both have to be stated**. One half is physics: a trailer holds about 19,958 kilograms. The other is this customer's measured flow: how many trailers a day actually run that lane. We shipped the failure that taught us this - somebody multiplied the real trailer weight by a "1.0 vehicles per day" nobody had counted, and the dimensionally-perfect result capped every unsized lane at one truck a day.

Two halves, from two different places. Both have to be stated.

PHYSICS - true of the world

19,958 kg per trailer
checkable, and true of every customer

YOUR FLOW - measured here

1.0 vehicles per day
nobody counted the trucks

One truck a day, on every lane nobody had sized. Nothing in the system complained.

What it cost, on a food manufacturer

Order fill stuck at 78% against a 99% ceiling, at any stock level, permanently - and the scorecard then read its own unreachable target as a reason to hold less stock.

Nobody hunting for hardcoded values ever finds this one, because half of it is genuinely correct.

So when we cannot measure your half the answer is **no limit at all**, not a small plausible one. An unconstrained lane is honest and visible; a lane capped at one truck a day looks like a considered decision and silently strangles the plan.

The second rule is that **settings get tested, not defaulted**. Our shipped default of fourteen days of cover should have been two and a half on one real network - cost fell by an order of magnitude with no loss of service. Nobody should copy two and a half; the transferable finding is the size of the gap between a plausible default and a tested one.

Seven questions worth asking any vendor

None of this is about a cleverer optimiser, which is why it is worth asking directly.

1. Which of my four planning jobs does your system actually do, and does it treat them as four decisions or as one plan refined four times?
2. Is my detailed plan required to reconcile back to my aggregate plan - and when it cannot, which one is treated as wrong?
3. What travels down between them - and specifically, is my plant scored against the S&OP number?
4. Which method decides each one, and what evidence would make you change it?
5. Where does my forecast's uncertainty come from, and what happens when my own history cannot support it?
6. Show me both halves of any number you are about to plan with.
7. Show me the test that set your default, not the default.

Powell's real contribution is not the four methods. It is the discipline of naming the decision, choosing a method for a stated reason, and keeping that choice open to evidence. *The Decision Factory* teaches that discipline rather than the mathematics, which is the harder and more useful half.

The Decision Factory (DeJans Jr. and Elam, 2026, foreword by Warren B. Powell) ships with *Decision Arena*, a simulation lab where students design and compete policies under uncertainty. The short-form version of the argument is [Optimizing the Uncertain](#) (Medium, February 2025).

See Autonomy in action

Walk through how Autonomy models, executes, monitors, and governs supply chain decisions with autonomous AI agents.

See It Live