

Predict the demand. Then decide *what to do about it.*

AI forecasting and mathematical optimization are two different questions with two different kinds of answer. Most mid-market planning tools attempt neither seriously; the few that do sell them as separate products. This article explains both, in plain terms, and shows how they run inside EAConnect Planning on the same tables — with every number traceable to its cause.

AUDIENCE	FORECASTING MODELS IN USE	OPTIMIZATION	READING TIME
Leaders of mid-size enterprises	Theta · ETS · Croston/SBA · Ridge AR	LP / MIP · HiGHS, SCIP, CP-SAT	15 min read

Series Hours, Not Months Evidence Forecast Studio demo · NorthStar S&OP test run · Optimizer user guide

Standard product claims only where shown; methodology as practitioner opinion

01 Two questions that get confused

A forecast answers *what is likely to happen*. An optimization answers *what should we do, given what is likely to happen and the limits we operate under*. Treating the second as a harder version of the first is the most common

mistake in planning software, and it is why so many "AI planning" projects produce a nicer chart and no better decisions.

PREDICTION

Forecasting

Input: history, calendars, known events.

Output: a range of likely outcomes per item per period. The quality measure is accuracy on data the model has not seen.

The failure mode is confidence without explanation.

PRESCRIPTION

Optimization

Input: a forecast, capacities, costs, rules.

Output: the set of decisions that minimizes cost or maximizes margin while honoring every rule. The quality measure is proof — that no better feasible plan exists. The failure mode is an answer nobody can audit.

The two are complementary and sequential. The forecast is an input to the optimization, and — this matters more than it sounds — the forecast's *uncertainty* is also an input. A demand plan that says "8,700 units, somewhere between 6,900 and 10,400" gives an optimizer what it needs to size safety stock. A single number does not.

What follows is a practitioner's account of both, grounded in how they are implemented in EAConnect Planning. Where the platform does something specific, it is stated specifically; where the point is general planning practice, it is offered as opinion from people who have built these systems for a long time.

Why this belongs together, in one platform

Forecasting and optimization share the same inputs (demand history, product master data, capacities) and the same consumers (the planning grid, the S&OP meeting, the board pack). Running them as separate products means reconciling two copies of the truth. Running them on one set of planning tables means the hand-off is a column, not a project.

02 What the forecasting engine actually runs

No algorithm selection, no parameter tuning, no data-science project. For every series the engine evaluates a fixed set of candidate models on recent

history it withholds from them, ranks them on how well they would have predicted it, and blends them across the horizon.

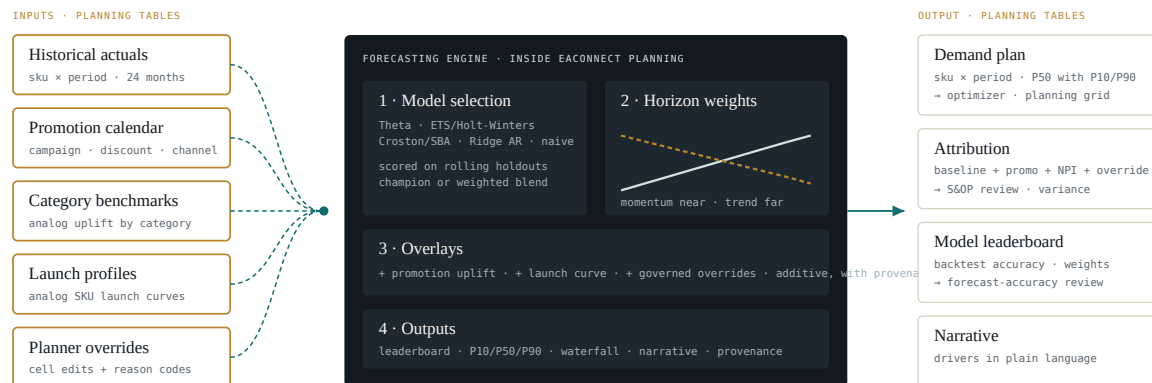


Figure 1. The forecasting engine as a pipeline. Every box on the left and right is an ordinary planning table; the engine in the middle is a service the platform calls. Nothing in it requires configuration by the planner.

The candidate models, and why these

The engine's candidates are deliberately classical. Each covers a demand pattern the others handle badly:

MODEL	WHAT IT CAPTURES	WHERE IT WINS
ETS / Holt-Winters	Level, trend and seasonality with exponentially decaying memory	Established products with a repeating annual shape — most of a consumer portfolio
Theta	Decomposes a series into long-run trend and short-run curvature and recombines them	Series with a clear direction but noisy months; a strong general-purpose baseline
Croston / SBA	Models intervals between demand events separately from demand size	Intermittent, "lumpy" demand — spare parts, B2B accounts that order a few times a year
Ridge AR	Regularized autoregression on recent lags	Near-term momentum after a step change, where seasonal models lag
Naive / seasonal naive	Last value, or last year's same period	The floor every other model must beat to earn a weight
The set favors models that are fast, stable on short histories, and interpretable. Deep-learning forecasters are excluded on purpose: on typical mid-market data volumes — two years of monthly history per SKU — they rarely beat this set on holdout accuracy and cannot explain their output.		

Holdouts, not hindsight

Each candidate is scored on *rolling holdouts*: the engine hides the most recent periods, fits on what remains, forecasts the hidden periods, and measures the miss. It repeats this across several cut-off points so the score reflects performance on data the model never saw. The champion in the example later in this article is ETS with a backtest accuracy of 89.2%; that number is the average of those out-of-sample tests, not a fit to history.

Weights that change along the horizon

A twelve-month forecast is not one problem. Months one to three are dominated by momentum — what happened last quarter is the best guide to next quarter. Months nine to twelve are dominated by trend and seasonality. The engine assigns weights step by step across the horizon, so near-term models carry the first months and trend models carry the later ones inside a single projection. Planners can choose a champion-only strategy (the best model everywhere) or a weighted blend; the default selects automatically.

The leader's question: "why should I trust an automatic choice?"

Because it is not automatic in the sense of hidden. Every run returns the leaderboard — which models were tried, how each scored on holdout, what weight each received. A planning manager can see, per product family, that ETS wins on beverages and Croston wins on spare parts, and that is exactly what a good analyst would have concluded by hand, several weeks later.

03 Overlays: the things a baseline cannot see

A statistical baseline knows only the past. Demand is shaped by decisions the past does not contain — a promotion not yet run, a product not yet launched, a commitment the model cannot know. The engine carries three overlays natively. Each is additive, each is auditable, and each can be switched on and off without touching the baseline.

OVERLAY 1

Trade promotions

Uplift is calculated, not guessed: from the category benchmark for the promotion type and from prior campaigns on analog products. The provenance is returned with the number — which analogs, which channel, what discount depth.

OVERLAY 2

New product introductions

A zero-history SKU gets a launch curve synthesized from an analog product's actual launch profile and scaled to the expected size. The new item enters the plan on day one with a defensible shape rather than a flat line.

OVERLAY 3

Planner governance

Any cell can be overridden. A reason code is mandatory, and the machine recommendation is retained alongside the override rather than replaced. The plan remains separable into "what the machine said" and "what people changed".

The third overlay deserves a leader's attention. In most planning tools an override erases the forecast it replaced, and with it the ability to learn whether the override was right.

Retaining both makes forecast-accuracy reviews honest: at quarter-end you can measure the machine, the planners, and the combination separately. Over a few cycles that tells you where human judgement is adding value and where it is adding noise — a question most organizations cannot answer today.

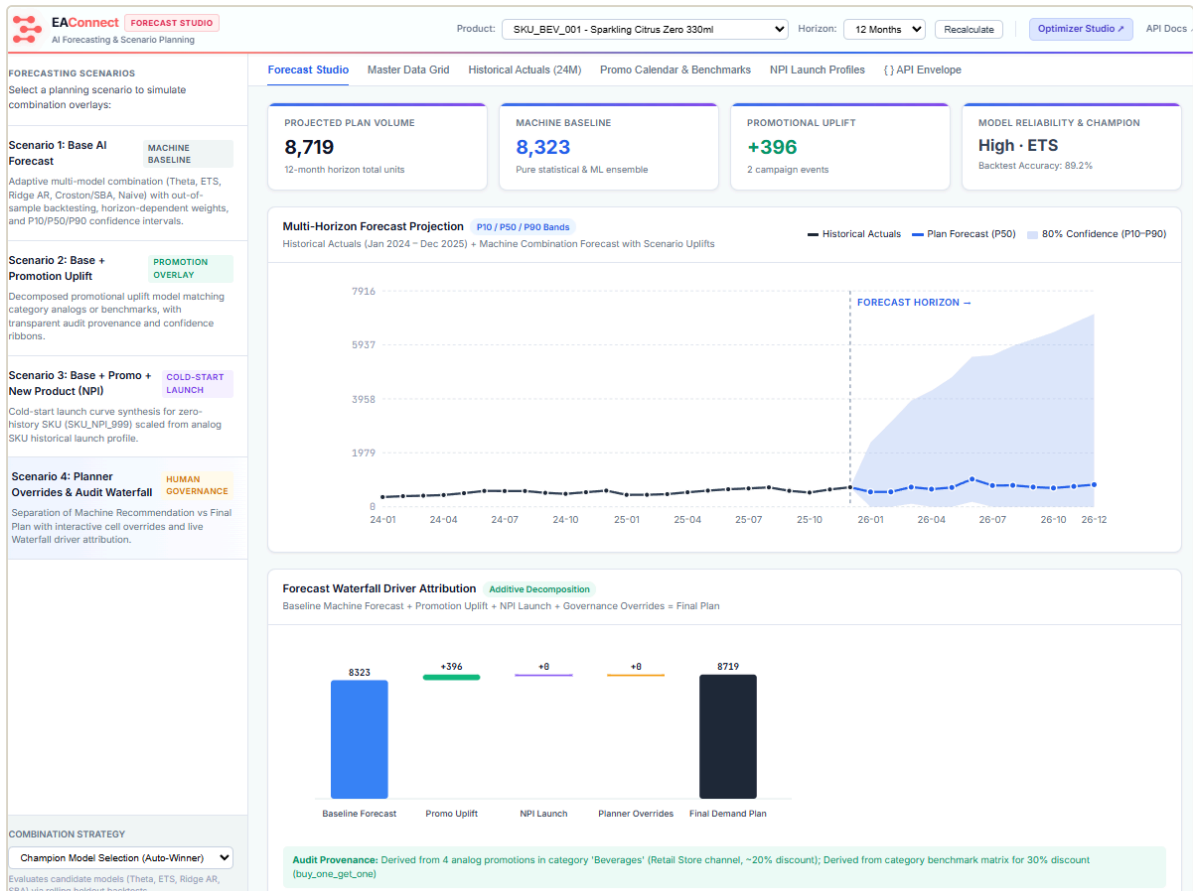


Figure 2. The Forecast Studio for one SKU over twelve months. Left: four scenarios on the same item — base AI forecast, base plus promotion, base plus promotion plus new-product launch, and the governed plan with overrides. Center: the projection with its 80% confidence band widening across the horizon. Bottom: the waterfall from machine baseline to final plan, and the provenance line for the promotion uplift.

04 Reading a forecast: four things every run returns

A forecast that cannot be interrogated will be overwritten. The engine returns,

with every projection, the four artefacts a planning meeting actually needs.

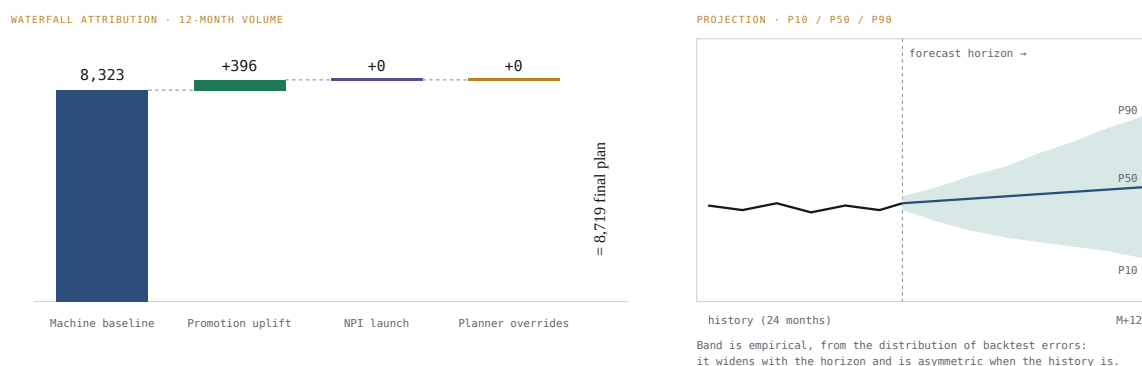


Figure 3. Left: the additive decomposition of the final plan in Figure 2. Right: the projection with its interval. Both are returned as data, not just drawn.

Leaderboard

Which models were tried, holdout accuracy for each, and the weight assigned. Answers "why this model".

Waterfall

Final plan = baseline + promotion + launch + overrides. Nothing hides in a residual. Answers "why this number".

P10 / P50 / P90

Intervals from backtest residuals, not a symmetric assumption. Answers "how wrong could this be" — the input safety stock needs.

Narrative & provenance

A plain-language summary of the drivers, plus the analogs, benchmarks and reason codes behind every overlay.

The practical consequence is that disagreement becomes specific. A sales director who thinks 8,719 is low can see that the baseline is 8,323, the promotion adds 396, and nothing else is in the number — and can argue about the promotion assumption rather than the whole forecast. That is a different meeting from the one most companies have.

05 The hand-off: uncertainty becomes a decision input

The forecast's interval is not decoration. It is what the optimizer needs to size buffers rationally.

A common rule in mid-market supply planning is a flat safety-stock percentage — fifteen percent of forecast, say, across every product. It over-protects the stable items and under-protects the volatile ones. With a P10/P90 band per item, the buffer can be set from the item's own forecast error: wide band, larger buffer; narrow band, smaller. In EAConnect the demand plan (P50) and its interval land in the same planning tables the optimizer reads, so the safety-stock target it honors is derived from the forecast's measured uncertainty rather than a fixed percentage.

What this means in practice

Working capital tied up in inventory drops on stable items and service level rises on volatile ones, from the same total stock. It is one of the few changes in planning that improves both sides of the trade-off, and it requires nothing more than using the interval the forecast already produces.

06 What the optimizer solves

Given a forecast, capacities, costs and rules, the optimizer finds the cheapest plan that honors every rule — and proves it. Inside EAConnect Planning it is a service that works on tables, with templates for the common problems and a model language for the rest.

Mathematically these are linear programs (LP) when every decision is a quantity, and mixed-integer programs (MIP) when some decisions are yes/no — open this lane, source this SKU from this plant, activate this supplier. The service compiles the problem from the uploaded tables and hands it to an open solver: HiGHS for LP and most MIP, SCIP for harder integer problems, CP-SAT for scheduling-type constraints. Which solver is used is a setting, not a project.

TEMPLATE

Transportation

Which plant or DC serves which customer, at what volume, at least cost within lane capacities.

TEMPLATE

Supplier allocation

Split purchase volume across suppliers under minimums, maximums, tiered prices and dual-sourcing rules.

TEMPLATE

Production planning

What to make where and when, under line hours, changeovers and inventory balance across periods.

TEMPLATE

Blending

Recipe or portfolio mixes that hit specifications at minimum cost.

Each template declares the tables it expects and ships with sample data, so a team can run a complete round trip before preparing its own. Problems outside the templates — the case in the next section is one — are written in the service's model representation and behave identically: the same validation, the same verification, the same explanation.

The workflow, from a planner's seat

1. **Upload tables** — once. They are typed, checked and stored as a dataset.
2. **Validate against the model** before any solve is spent: key uniqueness, types, nullability, cross-table rules.
3. **Solve**. Small problems return in seconds; large ones run as a job with progress, a notification on completion, and the option to cancel.
4. **Read the solved tables** — with the same key columns you uploaded, so joining the plan back to your data is a filter.
5. **Ask why**. Binding constraints ranked by shadow price, in plain labels. Re-plan under named what-ifs. If the plan is impossible, get told exactly which rules collide.

07 A worked case: a year of beverages

To exercise the service at a realistic scale we planned fiscal 2027 for a fictional company, NorthStar Beverages. The data is generated

deterministically; every figure here is reproducible from the archived run.

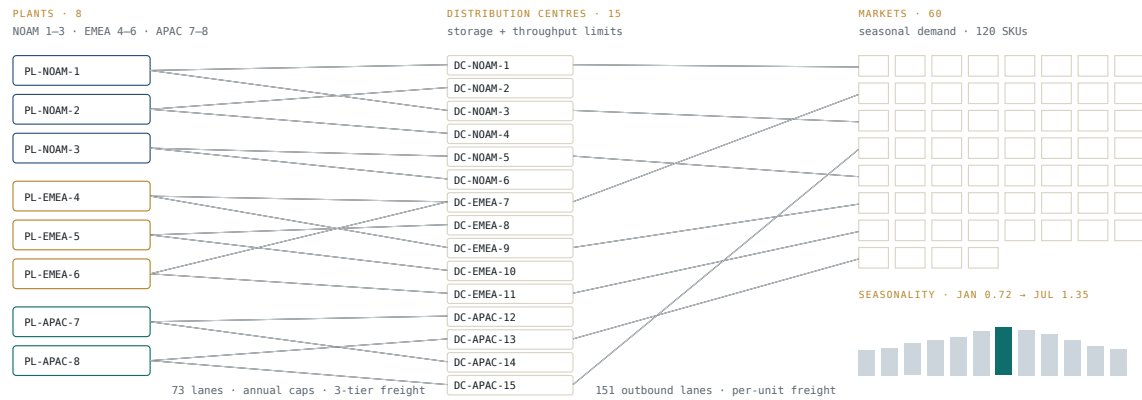


Figure 4. The NorthStar network. Cheap plants in APAC and NOAM, expensive plants in EMEA; one EMEA plant is fully down for maintenance in May. Twelve monthly periods, 29.9 million units of annual demand.

The decisions and the rules

DECISION	RULE IT MUST RESPECT	COST IT INCURS
Which SKU is sourced from which plant, for the year (295 admissible pairs, yes/no)	An active pair must meet a minimum annual commitment	Annual activation fee per pair
Monthly production per SKU per plant	Line-hours per plant per month, with maintenance dips; a maximum rate per SKU-plant	Unit production cost — APAC cheapest, EMEA dearest
Shipments plant → DC and DC → market	DC storage and throughput; only makers ship, only serving DCs deliver	Handling and outbound freight per unit
DC stock, month to month	Stock balance; a per-SKU safety-stock target (soft, with penalty)	Holding cost
Annual volume per plant → DC lane (73 lanes)	Annual cap per lane	Inbound freight on three-tier contracts — cheaper marginal rate as volume grows
Demand left unserved	Allowed	Shortfall penalty per unit, two to three times higher on premium SKUs
Thirteen input tables, 85,154 rows. Upload took 0.2 seconds; validation against the model 0.1 seconds. The objective is the standard one: minimize total cost for the year.		

How it was solved: the year first, then the months

The whole problem at once — 295 yes/no decisions on top of a 297,000-column linear program — produced no feasible solution in fifteen minutes. That is not a failure of the solver; it is the nature of the problem. So the test did what an experienced planning team does: it decided the year first, then planned the months, with the same model both times.

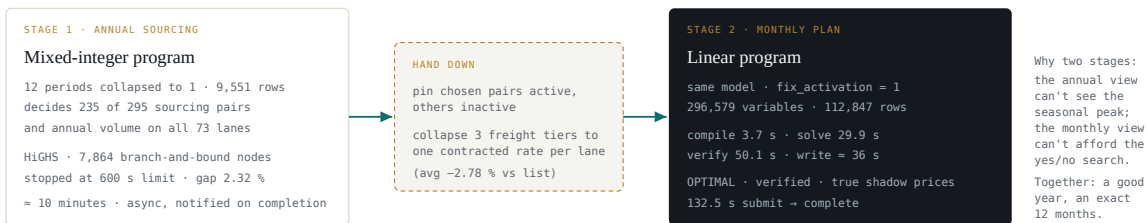


Figure 5. One model, two datasets, one parameter flipped. The stage-2 job is dominated by verification and writing results, not by the solver.

The result

32.55 M total plan cost, cost units · proven optimal	99.3 % fill rate · 29.69 M of 29.90 M units	235 / 295 sourcing pairs activated	16 / 16 independent business-rule checks passed
---	--	---	--

COST COMPONENTS · SHARE OF 32.55 M

Production	66.7 %
Inbound freight (contracted)	8.7 %
Outbound freight	8.4 %
Shortfall penalty	5.6 %
Activation fees	5.2 %
DC handling	3.2 %
Safety-stock penalty	1.1 %
Holding	1.1 %

Two-thirds of cost is production; freight in and out is 17 %; every penalty, fee and holding cost together is under a fifth.

PLANT UTILISATION · ANNUAL AVERAGE

PL-NOAM-3	99.0 %
PL-APAC-7	98.4 %
PL-APAC-8	93.9 %
PL-NOAM-1	86.5 %
PL-NOAM-2	85.4 %
PL-EMEA-4	72.5 %
PL-EMEA-5	68.4 %
PL-EMEA-6	41.4 %

Cheap plants run flat out; expensive EMEA plants absorb the rest. Shortfall with the season – 9,117 units in January, 37,181 in September – and is ze

Figure 6. Where the money goes and where the capacity is used. The plan's shape is what the data was designed to produce — which is the first test of whether an optimizer is doing something sensible.

What the shadow prices advise

A shadow price is the change in total cost from relaxing one constraint by one unit. It is the optimizer's way of saying where the plan hurts. Three signals from this run, in the order a leadership team would act on them:

- **Capacity at PL-EMEA-4 in May: –2,815 per line-hour.** May is the maintenance outage. One more hour there saves 2,815 in total cost — the strongest signal in the plan, and a direct case for moving the maintenance window.
- **PL-NOAM-3 in August and September: about –2,060 per line-hour**, roughly ten times the plant's unit cost. Peak-season capacity at the cheapest NOAM plant is the bottleneck. Overtime or a shift there pays for itself many times over.
- **62 of 295 minimum-commitment rules bind** — sourcing pairs producing only to honor the annual minimum. Those commitments are costing more than they earn and are the ones to renegotiate.

None of this required a modelling specialist to read. The constraints carry plain labels, and the explanation endpoint returns them ranked: "Capacity at plant PL-EMEA-4 in 2027-05".

08 Trust: how we know the answer is right

A leadership team should not have to take a solver's word. The service checks its own answer, an independent check rebuilds it from the source data, and two further exercises test whether it behaves like a plan rather than arithmetic.

FOUR LAYERS OF CHECKING

- 1 · Solver
reports OPTIMAL; primal and dual feasible; best bound equals objective
- 2 · Service verifier
re-checks every constraint against the solution before trusted: true · violations: none
- 3 · Independent check
pandas + source CSVs, no engine: 16 business rules · objective rebuilt to 2.3e-16
- 4 · Economic sense
five what-ifs and one deliberate infeasibility behave as a planner would predict

WHAT-IF SCENARIO SET · ONE REQUEST · Δ COST VS BASE

PL-EMEA-4 also down Jul–Aug	+3.4 %
NOAM demand +20 %	+7.3 %
Deactivate one SKU line	+15.8 %
Demand ×1.1 everywhere	+21.9 %
Drop market MK-EMEA-026	–2.3 %

Every delta has the sign and rough size a planner would predict. In the outage scenario plan pre-builds cola in January so DC stock covers the summer with no production at I

Infeasibility on purpose: remove the top SKU under hard demand → proven infeasible in 68-row diagnosis naming the SKU, and the capacity that would fix it (+30,326 units, I

Figure 7. Verification is layered so that a fault at any level — a solver tolerance, a compiler error, a misread coefficient — is caught by the level below it.

The third layer is the one to notice. It uses nothing from the service: only the source CSVs and a general-purpose data library. It recomputes demand satisfaction on every row, stock balance for all 21,600 SKU-DC-months, capacity, minimum commitments, tiered freight re-priced from the rate table, and the total cost from its eight components. The rebuilt total matches the service's to within floating-point noise. If the engine had misapplied a tier or

dropped a penalty, this is where it would show — and it is the check a finance team can repeat itself.

PHASE	WALL TIME	NOTE
Upload 85,154 rows	0.2 s	
Validate dataset against model	0.1 s	before any solve is spent
Stage 1 — annual sourcing MIP	608.7 s	time limit; gap 2.32 %
Stage 2 — monthly LP, submit → complete	132.7 s	solver 29.9 s of that
Sixteen independent checks	0.3 s	
Excel export, 326,445 rows	34.2 s	11.7 MB workbook
Infeasible run with diagnosis	100.1 s	
Five-scenario what-if set	500.0 s	
Entire test, end to end	27 min	measured in a test environment
Three minor findings were recorded, none blocking: the CSV loader treats the literal string NA as null; one status poll needed a retry; cancelling a running job waits for the verify and write phases to finish.		

09 Where each fits in the planning cycle

Forecasting and optimization are not a monthly ritual; they are steps in a cycle most mid-size companies already run informally. The value of having both in

the planning platform is that the steps share data instead of email.

CYCLE STEP	WHAT RUNS	WHO ACTS ON IT	OUTPUT LANDS IN
Demand review	Forecast refresh on new actuals; promotion and launch overlays updated	Demand planners, sales	Demand plan table with P10/P50/P90
Supply review	Optimizer run against the demand plan, capacities and costs	Supply and operations	Production, shipment and inventory tables; shadow prices
Pre-S&OP	What-if scenario set: outage, demand surge, dropped market	FP&A	Scenario deltas, row level
Executive S&OP	Waterfall, binding constraints, scenario comparison	Leadership	Decisions: capacity, commitments, service level
Financial plan	Volumes × prices and costs into the P&L model; working capital from inventory plan	Finance	Rolling forecast, cash forecast
The same tables carry through every row. The forecast that the optimizer planned against is the one finance prices — no re-keying between demand, supply and money.			

10 A practitioner's view for leaders

Opinions, from people who have implemented these systems at both enterprise and

mid-market scale. Not product claims.

Start with forecasting, not optimization

An optimizer amplifies whatever demand number it is given. A team that has not yet measured its forecast accuracy — machine versus planner versus combined — will optimize confidently against the wrong number. Two or three cycles of leaderboard and override tracking come first.

Optimization earns its keep on structure, not scale

The case above is large; most mid-market problems are not. A ten-supplier allocation or a single-plant schedule returns in seconds. What justifies the optimizer is not row count but the presence of real trade-offs — capacity versus service, contract minimums versus freight tiers — that spreadsheets settle by argument.

Insist on explanation as a condition of adoption

Any forecast or plan your planners cannot interrogate will be overwritten within two cycles, and the machine's value with it. Leaderboards, waterfalls and shadow prices are not analyst features; they are the mechanism by which the organisation keeps using the tool.

Use the interval, not just the number

The cheapest working-capital improvement available to most mid-size companies is replacing a flat safety-stock percentage with buffers sized from each item's forecast error. The forecast already produces the input; the change is a policy decision.

Do not buy these as separate products

Forecasting, optimization and the financial plan consume and produce the same tables. Separate products mean separate copies of demand, and the reconciliation between them becomes a standing cost that no one budgets for.

Where not to use them

Very short histories with no analogs; markets driven by a handful of large, negotiated deals; decisions whose constraints cannot be written down. In those cases a judgement-based plan with a clear owner beats a model, and the honest answer is to say so.

3,462 words · 15 min read