Causal AI Has Been Undeployable for 30 Years

Autonomy Platform · Public Blog

PERSPECTIVE

Technical May 2026

Causal AI Has Been Undeployable for 30 Years

Ari Joury's argument is right. Agents collapse the deployment cost. Here is what that looks like inside an industrial supply-chain substrate.

On May 11, Ari Joury (Wangari) published a piece in *Level Up Coding* titled *How Agentic AI Finally Makes Causal Inference Deployable*. The argument is narrow and precise: causal inference (Pearl's Structural Causal Model framework plus the do-calculus) has been a well-established methodology for 30 years, and it has been almost entirely undeployed in production. The math is not the problem. The deployment cost is. Joury decomposes the deployment cost into five stages, each historically requiring a specialist, and shows that agents collapse the cost of each stage while leaving the *judgment layer* with humans.

I want to back up Joury's claim from the inside, because Autonomy has been building exactly the substrate his argument implies for the last 18 months, and the framing he supplies makes it possible to say crisply what we have, what we are adding, and what the discipline is that makes it work.

What causal inference actually requires

A causal model is a Directed Acyclic Graph (DAG) plus a set of structural equations. Nodes are variables. Edges are directed causal relationships. The framework's key operation is Pearl's do-operator: `do(X = x)` represents an *intervention* that sets variable X to value x, severing its connection to its usual causes. The do-calculus is a complete set of inference rules that lets you compute the effect of such interventions from observational data, given a known causal graph. You can answer "what if we change X?" without ever having run the experiment. This is what makes the framework powerful.

This is also what makes it different from predictive ML, which estimates `E[Y | X, Z]` (the expected outcome conditional on observed features). The two estimands diverge whenever the action is non-randomly assigned, which is *always* in supply-chain decision-making. Promotions are scheduled in periods where the planner already expects elevated demand. Expediting happens precisely when an order is at risk. New product introductions get launched on products the marketing team selected for likely success. A predictive model trained on `outcome | action` confounds the action's effect with the population that gets selected for the action. The result, in each of those three cases, is a confidently-wrong answer that does not survive a CFO asking *"would that have happened anyway?"*

The five deployment stages, and where the cost lives

Joury's decomposition:

Stage	What it requires	Pre-agentic cost
1. Variable selection and domain scoping	Decide which variables enter the model	Expert workshops; slow, expensive, room-dependent
2. Causal graph construction	Specify which variables cause which	Constraint-based or score-based search produces a Markov-equivalence class, not a unique DAG
3. Graph validation and sensitivity testing	Test the DAG's implied independences; sweep over edge variations	Specialist statistician runs d-separation tests, refutation, sensitivity sweeps
4. Interventional query answering	Translate business question to do-expression; estimate	Specialist who knows both the business and the math
5. Audit trail and documentation	Defensible record of every modelling choice	Manual, post-hoc, incomplete

Each stage individually requires a different specialist. The cumulative cost has been the real reason causal inference has stayed in academic settings: not the math, the process.

What agents change

The agent's job at each stage is the *process* of producing the artefact. The human's job is judgment over the artefact. Joury is explicit that this is not "agents replace humans"; it is "agents handle process so humans can do more judgment."

- **Stage 1.** An agent reads the canonical literature, proposes a candidate variable set with citations. The expert prunes.

- **Stage 2.** An agent runs multiple discovery algorithms, surfaces disagreements between them as structured decisions. The expert resolves.
- **Stage 3.** An agent runs the test battery systematically, summarises results, flags violations. The statistician interprets the diagnostic report.
- **Stage 4.** An agent translates natural-language business questions to do-expressions, returns estimates with plain-language explanations. The user evaluates whether the translation matches their intent.
- **Stage 5.** Every agent action is logged with timestamp, inputs, outputs, and reasoning. The audit is a byproduct of the process, not a separate task.

Joury is also honest about what agents cannot do. Three caveats, each load-bearing: agents cannot determine causal direction from observational data alone (domain expertise is *load-bearing*, not optional); agents cannot validate their own translations (the human review at Stage 4 is the primary defence against a class of errors that are invisible in the output but consequential in the decision); agents are not a substitute for experimental data when the no-unmeasured-confounders assumption is violated.

Where Autonomy already sits

Pillar 4 of the Autonomy substrate is Causal AI. The mathematical infrastructure for the five stages is in place:

- A typed `CausalGraph` carrying nodes, edges, treatment, outcome, confounders, instruments, effect modifiers, and unobserved nodes you want to track even when you cannot measure them.
- Identification machinery that returns one of `BackdoorIdentified`, `FrontdoorIdentified`, `IVIdentified`, or `NotIdentified` from any registered graph, using do-calculus under the hood.
- A catalogue of estimators: Causal Forest, DR-Learner, Linear DML, Synthetic Control, Difference-in-Differences, Instrumental Variables, Mediation (front-door product-of-coefficients).
- A digital-twin counterfactual adapter for the cases where observational identification is weak: instead of borrowing other products as a synthetic control, the substrate replays the same product's structural counterfactual in the twin.
- A conformal-causal wrap (Lei and Candès 2021) producing P10 / P50 / P90 bands on treatment effects with valid finite-sample coverage, *regardless of whether the underlying causal model is correctly specified*.
- An override-effectiveness pipeline that measures the causal effect of every human override against the agent's counterfactual, updates a Beta posterior with per-update confidence, and feeds the result back into TRM training as a sample-weight signal.

What we have *not* had until now is the agentic process layer driving the five stages. Each registered DAG today is hand-authored by the TRM author for the decision class. Refutation runs per-call, not systematically. There is no NL-to-do-expression surface; all interventional queries are pre-computed inside the override-attribution pipeline. Joury's argument supplies exactly the missing framing.

The TRM-pretraining analogue: generic seed DAG library

Here is the architectural extension that the framing makes obvious. Autonomy already follows a *generic-then-specific* pattern in four places: synthetic-tenant curriculum then per-customer learning tenant; generic TRM pre-training then per-tenant fine-tuning; the Context Engine's base source registry then tenant-specific signal activation; the AWS Supply Chain Data Model then tenant extensions. Pillar 4 has been the fifth instance waiting to happen.

The generic seed DAG library is the Pillar 4 analogue of generic TRM pre-training. Three sources populate it:

Source 1: our own LP constraint formulations. Every constraint in our optimisation layer is a structural equation at the type level. When the solver code declares:

```
inv[t, p, s] = inv[t-1, p, s] + prod[t, p, s] - sum_d ship[t, p, s, d]
```

The free variables **`t, p, s, d`** are AWS SC Data Model entity-type indices, not instance values. The constraint says: *for any product at any site at any time, inventory evolves by this rule*. That is a generic structural equation. It holds for Coca-Cola at an Atlanta DC, for steel at a Pittsburgh mill, for ice cream at a Boston freezer, without modification. The DAG implied by that constraint reads **`shipments_in + production + prior_inventory + shipments_out -> inventory`**. Direction is

unambiguous (the LHS is endogenous, the RHS terms are causes). An extractor walks the constraint definitions and emits the implied DAG into the seed library. No LLM required; the agent extracts, it does not infer.

Source 2: canonical formulas from inventory and planning theory. Every formula in canonical OR/MS literature is a structural equation. EOQ: `` `Q* = sqrt(2DS/H)` `` is a 4-node DAG (demand, setup cost, holding cost -> optimal lot size). Safety stock: `` `SS = z(alpha) * sigma_LT * sqrt(L)` `` is a 4-node DAG (service level, demand variability, lead time -> safety stock). Wagner-Whitin, newsvendor, Little's Law, the Lee-Padmanabhan-Whang bullwhip amplification chain. An LLM agent extracts these mechanically because the formula syntax *is* the SCM syntax. Round-trip verification: a second agent re-derives the formula from the DAG; disagreements hold the candidate for human review.

Source 3: published causal frameworks for supply chain. The SCOR model, Forrester / Sterman system-dynamics DAGs (including the canonical bullwhip amplification chain), APICS / ASCM body of knowledge, 50 years of operations-research papers. This is Joury's Stage 1 applied to supply chain specifically: mechanical literature synthesis with citation tracing, producing candidate structural amendments to the DAGs extracted from Sources 1 and 2.

Sources layer additively. Source 1 gives the deterministic skeleton. Source 2 adds stochastic edges (variance propagation, service-level relationships). Source 3 adds behavioural and unobserved-confounder structure. Per-tenant override-effectiveness data then specialises the seed posterior the same way per-user posteriors partial-pool toward a group posterior in the existing pipeline. Every new tenant inherits the seed DAG. The TRM

author's job moves from *writing the DAG from scratch* to *reviewing and ratifying the seed*. That is exactly the cost shift Joury describes for Stage 2, applied at the substrate level.

Five opportunities for outcome attribution

The five-stage decomposition also gives a clean way to enumerate where agents earn their keep in the override-attribution pipeline specifically: the loop where we measure whether human overrides of agent decisions actually improve outcomes.

1. **Override-reason categoriser.** An LLM reads the operator's free-text override reason and proposes a structured confounder: `` `{type: competitor_pricing, source: external_alert, expected_direction: demand_up, time_window: next_2_weeks}` ``. When N overrides cite the same structured confounder *and* the matched-pair treatment effect moves in the implied direction, the substrate ingests the missing signal. This is the operational signal that closes the no-unmeasured-confounders failure mode Joury names as Caveat 3.
2. **Natural-language counterfactual queries in the Inspect surface.** An operator asks *"what would happen to fill rate if I forced safety stock 20% higher across Iberia?"* An agent translates to a do-expression against the decision's registered DAG, evaluates, returns the estimate. The agent's first move on any query is a disambiguation prompt because business questions are routinely ambiguous between `` `E[Y | ``

`X = x]` (conditional) and `E[Y | do(X = x)]` (interventional), and confidently answering the wrong question is Joury's named Failure Mode B.

3. **Counterfactual-strategy sanity check.** For each decision class, run both the naive counterfactual (substitute the action, re-score) and the digital-twin counterfactual (replay the same period with the action node disabled). When they disagree by more than a threshold, the substrate auto-promotes that decision class to the twin counterfactual. The naive substitution was missing dynamics only the simulator captures. Pure parametric, no LLM; closes a real attribution gap.

4. **Estimand staleness detector.** A daily background sweep runs Joury's Stage 3 sensitivity tests systematically (Manski / Rosenbaum bounds, conformal-coverage empirical check, `add_unobserved_common_cause` refutation). When the bounds widen, the estimand is flagged stale and refuses to feed the Strategist until recalibration. The substrate is honest about *when* the identification assumptions stop holding, not only honest *if* they hold at registration time.

5. **Causal-learning-event row on the Decision Stream.** When the override-effectiveness posterior shifts materially, an event row lands on the Decision Stream formatted to the AI·IO·ML Inspect contract: prompt (what evidence triggered the shift), decision (which estimand updated, by how much), expected (what behaviour change the substrate predicts), likelihood (conformal-calibrated coverage of the new estimate). The substrate's learning becomes inspectable per-event, not only auditable in aggregate. This is Joury's Stage 5 applied to learning events themselves.

Each of those carries an explicit *propose-not-commit* boundary. The LLM never writes to an active estimand, an active seed DAG, or a decision record. The commit goes through either a parametric mechanism (Bayesian posterior shift, conformal recalibration, causal posterior update) or a human curator. That discipline is what keeps causal AI deployable instead of letting LLMs silently corrupt the calibration story.

What the discipline buys

Joury closes with a sentence I want to borrow: *“the case for this architecture is not that it makes causal inference easy. It doesn’t. The case is that it makes causal inference viable.”*

The same sentence applies to industrial supply chains specifically. The substrate is not making the math easier. Pearl’s do-calculus is exactly as hard as it has always been. What changes is the cost of the process around the math. An agent that mines our own LP code for the type-level DAG closes Stage 2 for a tenant in minutes instead of weeks. An agent that translates a planner’s natural-language question into a do-expression closes Stage 4 in seconds instead of a week of statistician time. An agent that runs the daily sensitivity sweep closes Stage 3 for every registered estimand systematically instead of when something obviously breaks. An agent that emits a learning-event row whenever a posterior shifts closes Stage 5 contemporaneously instead of after the fact.

Stage 1 is where the agent assist matters most for a specific reason. The substrate’s hardest failure mode is silent unmeasured confounding: a planner overrode because they knew something the substrate did not, the substrate measures the override as “human judgement beat the agent,”

and the next training cycle up-weights overrides that should have been attributed to *missing data the substrate ought to ingest*. The override-reason categoriser turns the operator's tacit reasoning into a structured confounder candidate that the substrate can either ingest as a signal or surface as a gap. That is the closing move on the non-ignorability assumption the existing override-effectiveness pipeline is built on.

Where this sits in the stack

The Causal AI substrate is one of the four pillars on top of which the platform sits: AI agents, conformal prediction, digital twin, causal AI. The pillars compose: agents make decisions, conformal calibrates the uncertainty on each decision, the twin generates training data plus structural counterfactuals, and the causal layer attributes realised outcomes back to the decisions and overrides that produced them.

The framework deserves the respect of being extended into industrial supply chains, not contradicted. Joury's argument is the public statement that *causal inference is finally cheap enough to deploy in regulated industries*. The substrate it implies is what we have been building. The five-stage decomposition gave us the framing to talk about it crisply.

For the architectural depth behind this post, see the How Agents Learn whitepaper, and Ari Joury's original article on Level Up Coding.

See Autonomy in action

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

See It Live