From CRO to Purchase Journey Intelligence

Why most “conversion analytics” can’t explain conversion, and what we’re building instead (Part 1)

Conversion Rate Optimization (CRO) has a branding problem. Its real problem costs e-commerce a great deal more.

Here’s what’s costing them. CRO is often treated as a **UI tweaking discipline**: change colors, rearrange blocks, A/B test micro-variants, ship “best practices,” and call it progress. Meanwhile, the failures that drain the most revenue often sit elsewhere: **offer competitiveness, availability, trust, constraints, and the competitive game.**

Here’s the reframing:

> Conversion is not an “optimization” problem.
> It’s a **purchase journey feasibility** problem — and feasibility problems hide from optimization tools.

A purchase journey converts only if it’s (1) worth starting, (2) possible to complete, and (3) reliable under real-world constraints, not just on your best day.

That’s what we mean by **Purchase Journey Intelligence (PJI)**: a system that measures and explains the preconditions that make purchase journeys succeed or fail. This is the bet I made when I founded SHORA last year, and the rest of this series is what happened when I tested it.

Why this matters (and where it goes)

Most teams discover “conversion problems” too late, because revenue is the laggiest indicator in the building. By the time a conversion rate dip shows up clearly in dashboards, the underlying journey may have been broken for days, across devices and contexts, silently burning paid traffic and brand trust. These “partial” failures — too localised to move the macro KPI, too cross-cutting to belong to any one team’s dashboard — never show up on the radar of the analytics tools e-commerce teams already pay for, because the tools were never designed to look for them. In Avinash Kaushik’s terms, they are **micro-conversion breaks** invisible to the **macro-conversion** KPI.

The economics fit on a napkin. A mid-market online retailer with **€50M annual revenue** at a **3% baseline conversion rate** and a **€1,000 average order value** moves roughly **50,000 paid journeys a year**. A partial failure that drops conversion by **two absolute points** on **one third of**

traffic — a single broken delivery-eligibility check on mobile, in one country, for two weeks — bleeds roughly **€330,000**. That is a “small” bug.[1]

---

[1] The underlying e-commerce conversion benchmarks and order-value distributions are drawn from publicly available retail industry reports cited in the appendix of Part 2.

So the endgame isn’t “better dashboards” or “more experiments.” It’s a capability:

- detect journey failures earlier than revenue KPIs can,
- localize *what broke* and *where*,
- attach evidence engineers can reproduce,
- track reliability over time, not just averages.

That is where CROspector is going. Part 1 explains why.

Outcomes over activity: the measurement gap most analytics stacks were never built to close

Most analytics stacks are excellent at counting events and terrible at explaining outcomes. They were built for activity, not for accountability.

They can produce what Kaushik calls a **data puke** on demand: page views, click-through rates, bounce rates, funnel stage drop-offs, heatmaps, session replays, rage clicks. A team can present forty of them and conclude nothing.

But they often fail at answering the only questions that matter:

What exactly blocked this purchase?
Was the journey impossible, or merely unattractive?
Did the user fail because of friction, or because of constraints?
How often does this break, across time, devices, and contexts?

This is not a feature gap in your analytics product. It is a **measurement gap** — and it sits exactly where Kaushik’s **Trinity (Experience, Behavior, Outcomes)** says it would. Existing tools measure Behavior (clickstream). They report Outcomes (revenue) as a lagging KPI. They leave Experience uninstrumented. Every revenue dollar that bleeds through a partial failure bleeds in the gap the Trinity predicted.

When you measure the wrong thing, you optimize proxies. When you optimize proxies, you ship activity, not accountability — and activity is exactly what Kaushik’s *Web Analytics 2.0* was written to retire.

Why A/B testing on weak measurement becomes a treadmill

A/B testing isn't the enemy. Weak causality is. A team running rigorous A/B tests on top of weak measurement is doing the right discipline on the wrong substrate.

If your instrumentation can't explain why users fail, you end up in a loop: generate hypotheses (often UI-only because it's easiest), ship experiments, see negligible effects most of the time, explain the outcome as noise or lack of traffic, repeat.

LLMs do not fix this — they amplify it. What happens when you point a state-of-the-art language model at this exact problem, and how much it costs to learn that lesson, is the subject of Part 2.

Two videos that illustrate the gap

To make this concrete, here are two ways to "instrument" the same journey — the same purchase attempt, the same user, the same failure. The first is what most session-replay tools see today. The second is one rung better. Neither is enough to stop the revenue leaking, and the reason neither is enough is what the rest of this post explains.

Video 1: Clickstream telemetry (behavior exhaust).

Video 2: UI-grounded annotation (behavior bound to rendered UI geometry).

Clickstream telemetry (the common baseline)

A clickstream shows cursor movement and clicks as a path. It looks like data. But it's mostly behavior exhaust. It is Behavior in Kaushik's Trinity, and only Behavior — no Experience, no link

to Outcomes.

It is not bound to the UI elements that were actually visible. It doesn't encode state ("which step were we in?"). It doesn't encode semantics ("what was the user trying to do?"). It doesn't encode system responses ("what happened when they tried?").

So two people can watch the same squiggle and tell two different stories.

That's not intelligence. That's narrative.

UI-grounded annotation (a real step forward)

The annotated view overlays the cursor trace on the actual rendered UI and identifies screen regions/elements. This is immediately better because behavior is now bound to UI geometry. You can isolate candidate targets (CTAs, controls, modals). It's more falsifiable than a blank-canvas path. It is also still Behavior. It does not yet capture Experience or attach to Outcomes.

It still does not answer what the user was trying to do (intent), what the system responded (Experience), or whether the journey ever produced an Outcome.

In other words: the annotated view records **what the cursor did**. It does not record **what the system was supposed to do, and whether it did it**. Those are different categories of evidence, and Purchase Journey Intelligence is about the second — the one that catches the failures bleeding money.

The purchase journey is bigger than the UI — and that is where most of the lost revenue lives

A purchase journey is a sequence of states under constraints.

It includes friction. It also includes five failure modes that quietly cost e-commerce more than friction ever has:

- **Offer competitiveness** — your price loses to a competitor in the same shopping session.
- **Availability and eligibility** — the item is shown in-stock but is not deliverable to the buyer's address.
- **Trust and compliance** — the checkout asks for data the buyer is not willing to give.
- **Performance and reliability** — the same journey works today and breaks tomorrow on the same device.
- **The competitive game** — the buyer comparison-shops mid-journey and never returns.

CRO over-focuses on friction because friction is the easiest thing to see and to test. The higher-leverage failures sit in the constraints, not the UI. Every one of them is, in Kaushik's vocabulary, a failure of **Experience** that the existing **Behavior**-only stack cannot detect until the **Outcome** KPI has already collapsed.

A conversion rate is just an average over these hidden states. Averaging is how the failures stay hidden, and how the revenue keeps leaking.

Where CROspector comes from: measurement before optimization

CROspector starts with a simple premise:

> You can't optimize what you can't observe,
> and you can't observe what you don't measure as states, constraints, and evidence.

So we built an agent that probes purchase journeys on **public surfaces**, repeatedly, across time and contexts like a human would, and it records evidence.

Evidence. Not events.

We probe journeys in a non-intrusive, rate-limited way on public pages, designed to respect site stability.

The goal isn't to generate hypotheses. The goal is to produce verifiable facts about what makes a purchase journey succeed or fail.

In plain language: a record of **what breaks, how often, under which conditions, what kind of failure it is** (friction, constraint, trust, competitive position), and evidence the engineering team can reproduce. Not a dashboard. Not another data puke. A reliability record for the journeys that pay your bills — **actionable** in Kaushik's exact sense: a single click of the failure record gives the engineer the page, the moment, the state, and the reproduction.

Analytics vs journey intelligence

Here's the distinction:

Analytics answers: "What happened?"
Journey intelligence answers: "What happened, why, under what conditions, and how reliable is it over time?"

If you’re serious about conversion, you want the second because it changes what your organization can do. It shortens time-to-detect, reduces time-to-diagnose, and prevents teams from spending months optimizing proxies while a constraint failure silently bleeds revenue.

Where we’re heading (Part 2 teaser)

Part 1 is about the reframing and the measurement gap.

Part 2 is the receipt — what happened when we tried to make a state-of-the-art language model produce the Experience layer of Kaushik’s Trinity, at scale, across millions of live retailer pages. It did not work. The accuracy rate was 9%. Part 2 explains the AWS invoice, what we expected, and what the data forced us to admit.
