The Value and Price of Understanding

Sameer Gupta

2026

xameer@zohomail.in • +91 799 311 0170 • Ananthapur, India

— — —

There is a question I have been stuck on for nearly two decades. Why do some systems stay understandable as they grow, while others just — collapse? Not break in an obvious way. Collapse into complexity nobody can reason about anymore.

Every domain I have worked in has been a version of that question. Statistical models, type systems, distributed infrastructure, product strategy, smart contracts. The surface looks different each time. Underneath it is the same thing: I want to know where the abstraction leaks and why.

It started with a failure I could not explain.

IIT Bombay, second year. I built JeeCarnot out of a dorm room — statistical rank projections for JEE aspirants. It worked. Students used it. The value was real. What I could not crack was the business, and the reason turned out to have nothing to do with the product.

This was before UPI. Getting paid meant going through telecom VAS channels. Airtel's revenue share would have left almost nothing for the company. Reliance wanted to acquire the product in a way that cut me out entirely, while the core model was still changing. The only clean path to revenue was selling user data. I didn't do that. The company closed.

What bothered me afterward wasn't the loss. It was that I had built something whose logic was sound — and it had run straight into an environment whose logic I didn't understand at all. I had no way to even formally describe the gap, let alone reason about it.

So I went looking for languages that could.

Lambda calculus. Type theory. Denotational semantics. Formal models of concurrency. No course, no supervisor, no research group. I read papers and books on my own, worked through proofs by hand, and when an argument felt shaky I pulled it apart and rebuilt it until it held. The bar I set for myself wasn't *can I use this in a job* — it was *could I derive this from nothing if I had to*.

By any normal measure this was a bad use of time. No credentials came out of it. No portfolio entries, no papers, no PhD. An academic path into formal methods without institutional support in that environment was not something I could invent out of nothing, so I didn't try. What I had was the work and whatever it built in me.

Years later, at Haskledger, I wasn't learning GADTs and phantom types under deadline. I already knew them. I owned the proposal, the product direction, the milestone delivery. The denotational interpreter, the mapping of DSL terms to Plutus Core via rewrite rules, the compile-time resource constraints — these weren't new ideas I was picking up. They were things I'd been sitting with for years, finally applied at scale. The funded proposal and the sidechain cost reductions came out of that. They didn't produce the obsession. The obsession produced them.

The same thing shows up in infrastructure, just differently. When etcd loses its encryption keys, or a CNI binary path silently breaks in NixOS NFTablesMode, or ArgoCD stalls on a taint nobody documented — what gets you through without data loss isn't knowing the procedure. It's having a precise enough model of what the system is *supposed* to be doing that you can see exactly where reality has diverged. You can't fake that with experience alone. You need to have actually thought hard about the formal structure underneath.

The hardest thing I've done wasn't any single project. It was spending years building a way of thinking — about systems, about abstractions, about where they hold and where they give — without knowing whether it would ever have a commercial return. I kept going because the alternative was worse: working permanently at the surface of systems I didn't really understand.

That's still what I'm doing. The question hasn't closed. I don't expect it to.