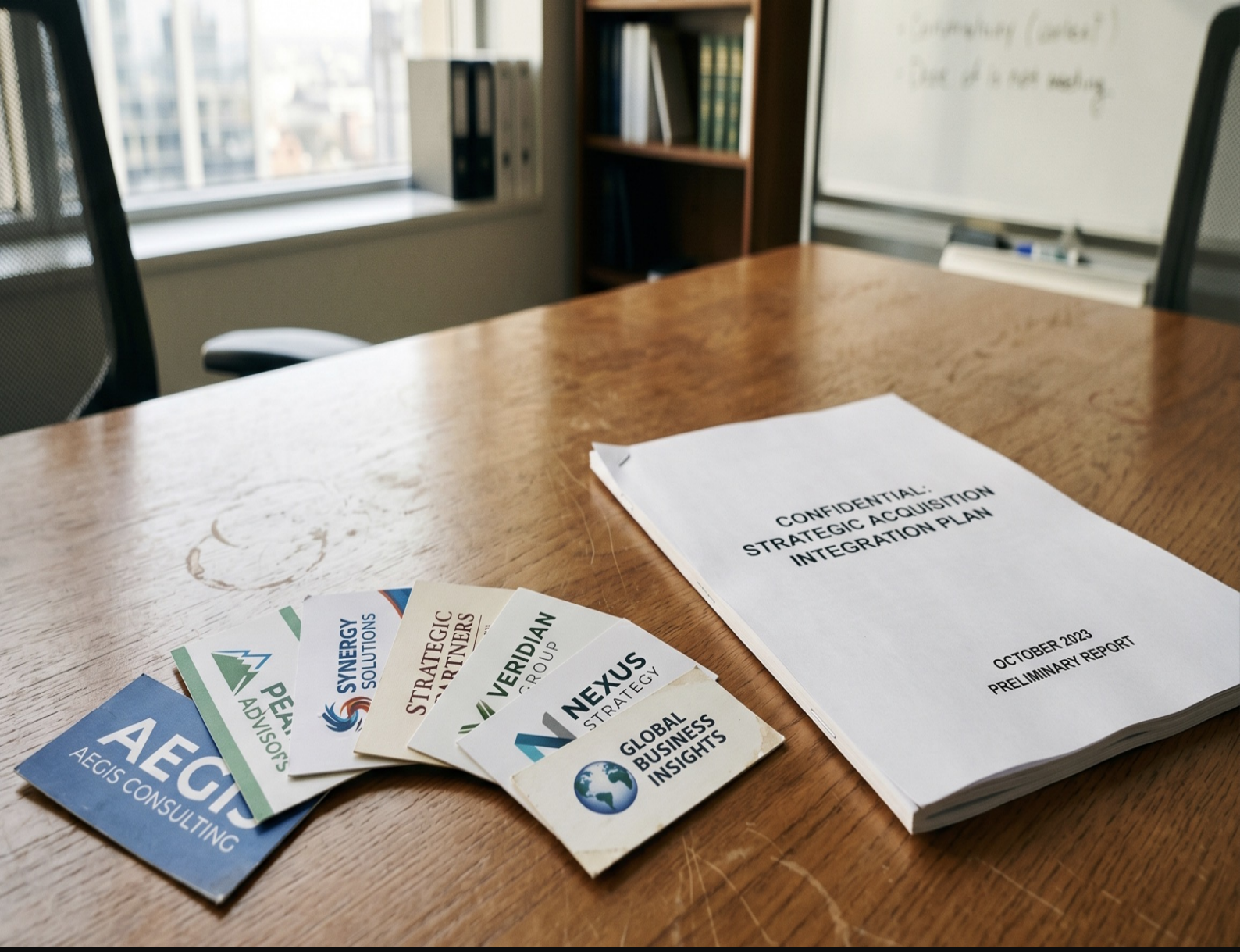

INVESTPUPPY UNVARNISHED

InvestPuppy

Complexity as cover: The consultant layer

*For anyone else who's been in **that** room.*

Serious about outcomes. Not serious about ourselves.

About InvestPuppy

InvestPuppy builds Vektor, a systematic portfolio management platform for wealth management professionals.

What this series is

Every experienced practitioner thinks it. Almost nobody publishes it: *there has to be a better way.*

We thought it long enough that we went and built one.

Before that, we spent decades in the rooms.

The same project was failing for the third time. The project plan had been signed off before anyone had understood the first requirement.

The relationship was too warm for anyone to say so.

It isn't incompetence.

It's the incentive structure doing its job perfectly.

Nobody planned it this way. Nobody needed to.

We got frustrated enough to write it down. Then frustrated enough to build something. These papers are the writing-it-down part.

The something we built is a separate conversation.

The consultant's primary product is not advice. It is a document. The document exists to justify the advice, to evidence the engagement, and to ensure that when the project fails, the advice was correctly recorded. The document does not make the project succeed. It makes the failure legible.

1. The Structure of the Relationship

There is nothing dishonest about the consulting relationship as it is typically structured. The vendor sells software. The client buys software. Between them, one or more consulting firms are engaged to manage the implementation. The consulting firms charge by the day. This is where the incentive problem begins — not with bad faith, but with arithmetic.

A ten-week implementation produces one invoice. A twenty-week implementation produces two. The consulting firm does not plan for the project to extend. It does not need to. The conditions that produce extension are already present in the structure of the engagement.

> The question is not whether the consultant wants the project to succeed. Most do. The question is whether the conditions under which they are paid require the project to succeed. They do not. They require the project to continue.

> No practitioner, managing live client portfolios while attending project workshops, has ever thought: what would really help now is a few more cleverly constructed slides. They need a working system, configured for how they actually operate. The consultant's deliverables are not designed to produce that. They are designed to evidence the engagement.

2. When There Is More Than One Firm

The single-firm engagement is the idealised case. The real case is more complicated: a systems integrator managing the technical implementation, a strategy firm that scoped the transformation, and a change management practice running the adoption programme. Three firms, three commercial relationships, three sets of incentives — none of which are aligned with each other, and none of which are fully transparent to the client.

When something goes wrong in a multi-firm engagement, each firm's first move is to locate responsibility in another firm's workstream. The client watches three parties explain why it is someone else's fault while the project continues to fail. Nobody is accountable for the outcome, only for their workstream. The project belongs to all of them. Which is to say, it belongs to none of them.

3. Complexity as a Product

Complexity is not an inevitable feature of financial software implementations. It is, in many cases, a product — produced and maintained by the parties who benefit from its existence.

The consultant who simplifies is the consultant who reduces their own engagement. This is not a character flaw. It is a rational response to a rational incentive. The implementation that could have been delivered in eight weeks with three people becomes the implementation delivered in twenty-four weeks with nine, because the specification was constructed in a way that made the longer engagement the only legible path.

> The warmth of the consulting relationship is the mechanism by which this asymmetry is maintained. A trusted adviser is harder to challenge than a vendor. Asking whether the project could be simpler feels like asking whether the advice was good.
>
> The relationship was built to prevent that question from being asked comfortably.
>
> The complexity belongs to the engagement. Which means it belongs to no one.

4. The Legibility Problem

The client's inability to evaluate the specification is not stupidity. It is a structural feature of the market. Financial software is complex. The people who understand it in detail are, by definition, the people who have been working in it — which means the vendor and the consulting firms. The client is buying expertise they do not have. This is the correct reason to engage a consultant.

The problem arises when the information asymmetry that justified the engagement is then used to extend it. The client cannot challenge a timeline they cannot evaluate. They cannot simplify an architecture they cannot read.

The warmth of the consulting relationship is the mechanism by which this asymmetry is maintained. A trusted adviser is harder to challenge than a vendor. Asking whether the project could be simpler feels like asking whether the advice was good.

The relationship was built to prevent that question from being asked comfortably. In a multi-firm engagement, each relationship performs this function independently. Nobody designed it this way. Nobody needed to.

The complexity belongs to the engagement. Which means it belongs to no one.

5. What a Better Engagement Looks Like

A consulting engagement structured around outcomes rather than duration looks different from the outside. The project plan is shorter. The workstreams are fewer. The specification is written in language the client can read and challenge. Milestones are tied to functional delivery, not document production.

Three questions worth asking before the engagement begins:

One. Is the project plan written in language our operations team can evaluate without assistance?

Two. Are the milestones defined by what the system does, or by what documents have been produced?

Three. If we are engaging multiple firms, which single entity is accountable for the outcome — and what happens to their engagement if the outcome is not

delivered?

The consultant layer is not the problem. An engagement structure that rewards complexity and diffuses accountability is the problem.

They are not the same thing.

The Unvarnished Series

Thirteen field notes on why financial software implementations fail.
Published under our own name. All freely available, no registration required.

Why Implementations Fail (UNV-01)
The same implementation failures repeat across the industry, unrecorded, because no party benefits from writing them down. This paper names the pattern openly, once, rather than let it repeat quietly again.

Nobody Owns It (UNV-02)
Ask a vendor who owns an implementation and they say the client. Ask the client and they say the vendor -- both are right, and that shared, undefined ownership is the actual failure mode, not any single decision either side made.

Complexity as Cover (UNV-03) (you are here)
A consultant's real product is often the document justifying the advice, not the advice itself -- built to survive the project's failure, not prevent it. Complexity becomes the cover story for that gap.

Plan Last (UNV-04)
Project plans are written before anyone understands the workflow they claim to schedule, in columns spanning weeks one through thirty-six. Planning first and discovering later gets the sequence backwards.

The Change Request Trap (UNV-05)
A change request is a formal admission that the original specification was wrong, and also a commercial mechanism for charging separately for what should have been included from the start. Neither party is surprised when it happens.

Invisible Stakeholders (UNV-06)
Every implementation has a second set of stakeholders who never attended a workshop or reviewed a specification, yet are the ones who have to live with what was built. Their absence from the room doesn't make them absent from the consequences.

The Hammer, the Tool, and the Methodology (UNV-07)
A tool gets picked up, used for the job it suits, and put down. A methodology requires certification, governance, and a steering committee -- and the industry keeps reaching for the second when it only ever needed the first.

Life After Live (UNV-08)
Go-live is the final milestone on the Gantt chart, but it's the beginning of the harder problem: an organisation built around a legacy system now has to actually live inside the new one. The project plan ends; the real work doesn't.

The U in UAT (UNV-09)
UAT sign-off documents carry the right signatures -- from consultants testing scripts written by other consultants, not from anyone who actually uses the system day to day. The U in UAT is the part everyone skips.

Nine Women (UNV-10)
Fred Brooks named it in 1975 and the same boardroom conversation still happens fifty years later: a project is late, so the proposal is more people. The logic feels intuitive and has always been wrong.

The AI Revolution — Automating Bad Practice at Scale (UNV-11)
AI doesn't fix a broken process -- it executes it faster, at greater scale, with fewer chances for the human hesitation that might have caught the problem. The real question isn't whether an AI implementation will work, but what it will end up working very hard to get wrong.

Model Bank — the perfect solution for banks that don't exist (UNV-12)
A 'model bank' configuration is a composite of institutions that don't exist, built from someone else's defaults before your institution was ever part of the conversation. Every subsequent customisation is just correcting an assumption that was never really about you.

The parameter that broke the camel's back (UNV-13)
A platform demo answers every question with yes, right up until one specific parameter turns out to be the one nobody actually tested. This paper is the account of exactly where that line sits, and what breaks on the other side of it.

Our Better Way

This is what we built instead.

VEKTOR by INVESTPUPPY

"So, what did you do about it?" — a fair question after thirteen papers on what goes wrong. InvestPuppy builds Vektor, a systematic portfolio management platform for wealth management professionals — the direct answer, series by series.

These four papers are the how to the thirteen whys above — not a sales pitch, but a technical account of the specific choices Vektor made where the industry's default pattern would have produced the same failures just described.

– Technical Debt (BW-01)
– Vektor House Implementation (BW-02)
– The Configurability Gap (BW-03)
– AI as Instrument (BW-04)

More at investpuppy.com