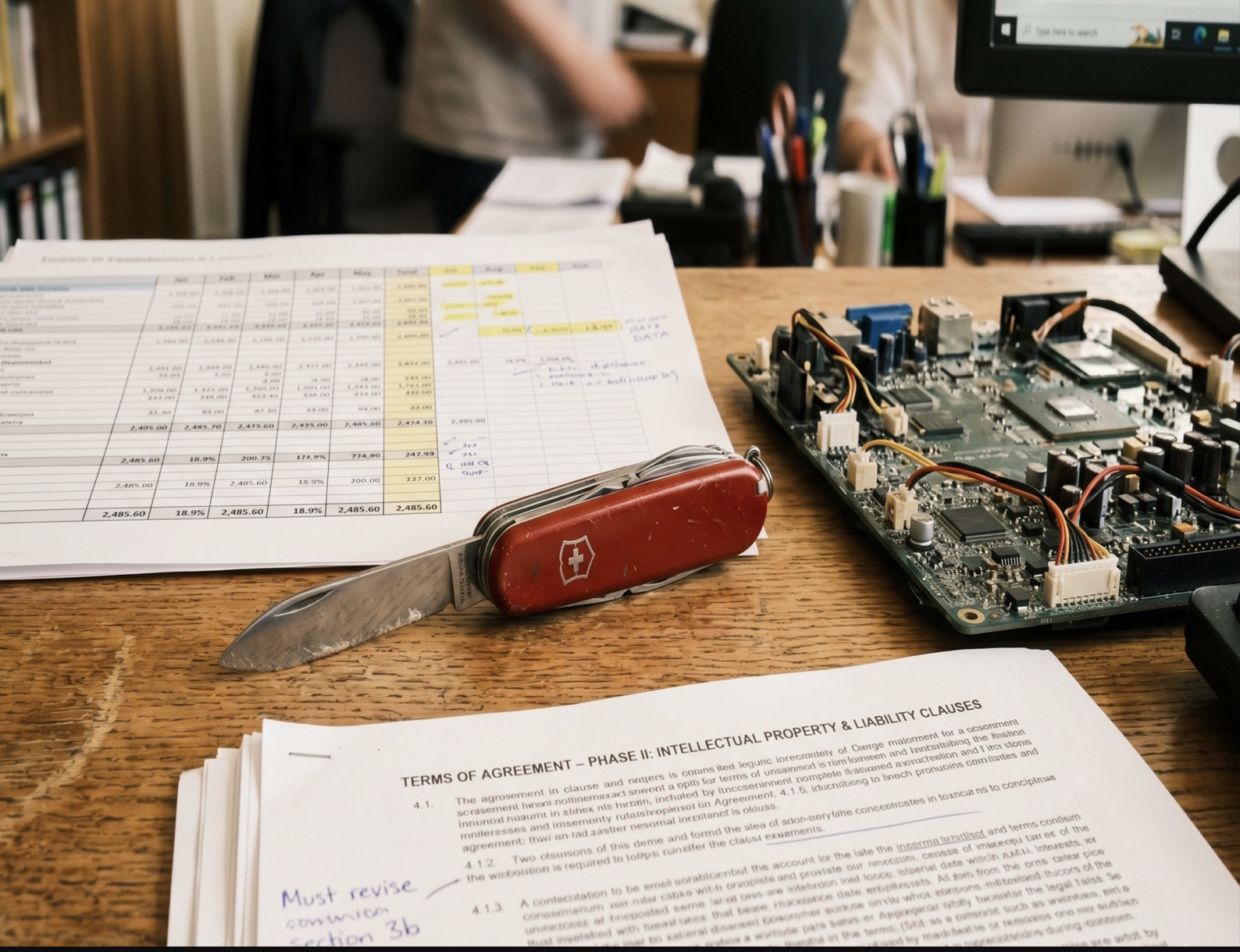

INVESTPUPPY UNVARNISHED

InvestPuppy

A hammer is a tool, not a methodology

*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.

> ***A tool is something you pick up, use for the task it is suited for, and put down. A methodology is something that requires certification, governance, a steering committee, a maturity model, and a two-day offsite. The gap between these two things is not semantic. It is the gap between something that serves the work and something that the work is made to serve.***

The tool became a methodology when there was money in the methodology.

1. How Tools Get Elevated

The elevation of a tool to a philosophy follows a consistent pattern. The tool is genuinely useful — it solves a real problem in a specific context. Early adopters use it well, for that problem, in that context. Results are good. Case studies are written.

Then something shifts. The tool acquires a name. The name acquires a capital letter. Consultants begin building practices around it. Certification programmes are created. Organisations begin asking not 'does this tool fit our problem?' but 'are we doing this correctly?' The question of fit is replaced by the question of compliance.

2. Six Tools That Became Methodologies

Agile.

Agile began as a set of principles for software development teams working on problems too complex to specify in advance. It worked, in that context. It then became a universal project philosophy. Organisations began applying Agile to regulatory compliance programmes and financial software implementations where the work was not complex in the Agile sense — it was simply uncertain.

Design thinking.

A creative problem-framing tool applied to regulatory compliance challenges. Applied outside its context, it consumes time and produces empathy maps for problems that needed process maps. Wrong context.

Blockchain.

Between 2017 and 2020, a significant portion of financial services strategy documents contained the word 'blockchain'. Most no longer do. The methodology phase has, mercifully, passed. This is what the dust settling looks like.

Data-driven decision making.

The corrective was necessary. The correction became the religion. Organisations that could not approve a budget without a dashboard. Metric gaming. The confusion of correlation with insight.

JIRA and tooling as methodology.

The project management tool becomes the project management methodology. The tool determines the work structure, the priority-setting process, the progress reporting cadence. The tail wags the dog.

AI and machine learning.

The term 'AI-powered' has become a marketing claim so widely applied that it has ceased to carry information. The tool is genuinely transformationally powerful for the tasks it is designed for. The elevation is happening now. The dust has not settled.

3. On AI Specifically

AI is a powerful tool. The capabilities are real: pattern recognition at scale, anomaly detection, systematic optimisation across large parameter spaces. For these tasks — tasks where the problem is defined, the data is available, and the output can be evaluated objectively — AI performs at a level that human analysts cannot match consistently over volume and time.

It is also the wrong tool — not merely imperfect, but genuinely unsuited — for tasks that require contextual judgment, accountability, relationship management, and regulatory interpretation.

> When we built our own portfolio management platform, we made these decisions explicitly before writing a line of code. Systematic methods — signal selection, portfolio optimisation, lot sizing — are pattern-recognition and calculation problems. We used them. Client judgment, mandate interpretation, relationship management, and the recognition that something has changed that the data hasn't yet captured — these are not pattern-recognition problems. We did not use systematic methods there. The platform works because the boundary was drawn first.

4. Using Tools as Tools

A tool used correctly has a defined scope: the tasks it performs, and the tasks it does not. A capable tool applied to the wrong problem produces the wrong result efficiently, at scale, and with the confidence of a methodology behind it.

Three questions before deploying any tool as a methodology:

One. For what specific tasks is this tool the best available option? Name them precisely.

Two. For what tasks is this tool being proposed that it was not designed for?

Three. Who benefits from the deployment of this tool as a methodology — and is their benefit aligned with the quality of the outcome?

> Three. Who benefits from the deployment of this tool as a methodology — and is their benefit aligned with the quality of the outcome, or with the duration of the engagement?

Pick it up. Use it for what it does.

Put it down.

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)
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) (you are here)
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