INVESTPUPPY UNVARNISHED

InvestPuppy

Plan last — discovery before configuration

*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 project plan is almost always the first formal document produced in a financial software implementation. It describes, in columns, what will happen in weeks one through thirty-six. It is produced before anyone has understood the workflow it is meant to govern. This is not a planning error. It is a procurement requirement. The client asked for a plan. The vendor produced one. The project was signed.

1. The Sequence Problem

There is a correct sequence for implementing financial software. It is: discover, then design, then configure, then test, then deliver. Most implementations do not follow this sequence. Most implementations follow a different one: plan, then configure, then discover, then redesign, then raise a change request, then extend the timeline, then retest, then deliver late.

The second sequence is not produced by incompetence. It is produced by a procurement process that requires a project plan before a discovery process has been completed. The client needs a timeline and a cost before they can approve the budget. The vendor needs a signed contract before they can begin the work. The project plan is the document that connects these two requirements.

It is, necessarily, a work of fiction.

2. What Discovery Actually Is

Discovery is not a workshop. It is not a series of stakeholder interviews documented in a requirements spreadsheet. It is not a two-day session in a hotel conference room that produces a one-hundred-slide deck and a 'discovery complete' milestone.

Discovery is the process of understanding, in sufficient detail, how the organisation currently works — including the workarounds, the exceptions, the manual processes, the data quality problems, and the things that are done

by one person who has been doing them for eleven years and has never written them down.

> The organisation that knows exactly what it needs from a new system does not need a discovery process. That organisation does not exist. Every organisation that has implemented financial software has discovered, during or after the implementation, requirements it did not know it had. Discovery does not create these requirements. It finds them before the configuration does.

3. The Configuration Trap

Configuration begins before discovery is complete because the project plan requires it to. Week four is the start of configuration. It says so in the plan. The discovery findings arrive in week three. Some of them are incompatible with the configuration decisions already made in weeks one and two.

This is not a project management failure. It is the project plan doing exactly what it was designed to do — providing a structure that allows the project to proceed before the information needed to proceed correctly is available.

> *A system configured before the workflow is understood is not a configured system. It is a system configured for the workflow that was assumed. The distance between the assumed workflow and the actual workflow is the scope of the rework.*
>
> *Every implementation has this distance. Most implementations discover it in production.*
>
> ***The board should have been filled before the first screen was touched.***

4. The Correct First Artefact

The correct first artefact in a financial software implementation is not a project plan. It is a workflow map — an accurate description of how the organisation currently processes the things the new system will process.

Three things a real discovery process produces:

One. A workflow map that the operations team recognises as accurate — not the process as it was designed, but the process as it is actually run.

Two. A documented list of exceptions, workarounds, and edge cases that the system will need to handle, provided by the people who handle them daily.

Three. A set of configuration decisions made before configuration begins, with the rationale recorded — so that when assumptions prove incorrect, the correction is made against a documented baseline, not against memory.

Plan when you know what you are planning.

Not before.

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) (you are here)
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