INVESTPUPPY UNVARNISHED

InvestPuppy

The change request trap

*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 change request was not unexpected.

> ***A change request is a formal acknowledgement that the original specification was wrong. It is also a commercial mechanism — a way for the vendor to charge for the work that should have been included in the original scope, and a way for the client to request the thing they needed but did not know how to ask for. It is not a sign that something has gone unexpectedly wrong. It is a sign that the project was specified before it was understood.***

It was inevitable from the day the specification was signed.

1. The Anatomy of a Change Request

Every change request has a before and an after. Before: a specification that described what the system would do. After: a discovery that the specification was incomplete, incorrect, or based on an assumption that turned out to be false.

The commercial context of this discovery matters. The power dynamic at change request stage is not the same as the power dynamic at procurement stage. When the specification was being written, the client had choices. By the time the first substantive change request arrives, the client has invested in the relationship, committed to a go-live date, and is in no position to walk away. The vendor knows this. The change request is priced accordingly.

2. The Volume Problem

A single change request is a correction. Five change requests is a pattern. Ten change requests means the specification was not fit for purpose.

Most implementations with significant CR volumes reach that volume not through a series of genuinely unforeseeable events, but through the systematic

underspecification of a small number of critical areas: integration points, exception handling, and reporting requirements.

> None of these are surprises in retrospect. They are the same underspecified areas in every implementation. The vendor has seen them before. The consulting firm has seen them before. The client has not. The asymmetry of experience is one reason these areas remain underspecified: the party with the most incentive to specify them thoroughly is the party least likely to flag the gap before the contract is signed.

3. The Contract Behind the Change Request

The change request mechanism does not operate in isolation. It is supported by the contract. A well-drafted vendor contract places the risk of specification incompleteness on the client: the vendor delivers what was specified, and if what was specified is not what was needed, that is a scope change. Scope changes are change requests.

What many clients do not examine carefully enough at contract stage is the definition of 'specification'. In most vendor contracts, this definition is limited to the signed document. Verbal representations made during the sales process, demonstrations of functionality, and commitments made in pre-contract workshops are explicitly excluded.

> *The change request is not a failure of process. It is the process. The original specification was always going to be incomplete. The vendor knew this. The consultant knew this. The client suspected it but could not say so.*
>
> *The CR mechanism is the commercial structure that converts that incompleteness into revenue. It does not need to be dishonest to be effective.*
>
> ***It only needs to exist.***

4. Reducing CR Exposure

CR exposure is not eliminated by a tighter contract. A tighter contract produces a more defensive vendor, a more adversarial relationship, and the same change requests priced differently. CR exposure is reduced by completing the discovery before the specification is written.

Three mechanisms that reduce CR volume in practice:

One. Specification sign-off by the people who will use the system, not only the people who commissioned it — including explicit sign-off on the handling of the twenty most common workflow exceptions.

Two. Integration mapping completed and agreed before the contract is signed, with third-party system owners included in that process, not added later.

Three. A contract definition of 'specification' that explicitly includes demonstrated functionality, pre-contract commitments, and any functionality referenced in the sales process — and a vendor willing to sign it.

Every change request is a discovery session that happened too late.

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) (you are here)
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