DESIGN DISCIPLINE

Configure first: the quiet discipline of a clean core

Customisation feels like getting exactly what you want. It is also a standing liability you agree to carry at every upgrade, forever. The discipline is knowing the difference.

Roltrader Consultancy Group · 6 min read

There is a seductive logic to customising an ERP. The standard process is 90% right; the last 10% is how your business is special; a developer can close the gap in a sprint. Each individual customisation is defensible, even sensible. The problem is never the first one. The problem is the two thousandth, and the fact that by then no one remembers why the first four hundred exist — but the system upgrade can't proceed until every one of them has been re-tested against the new release.

"Clean core" gets dismissed as vendor dogma, a way to sell you the standard product and bill you for the gaps. That cynicism misses what the discipline is actually protecting. A clean core is not about denying yourself capability. It is about keeping the system in a state where you can still adopt the improvements you are paying an annual fee to receive.

Customisation is a liability you agree to carry

When you configure within the standard — flags, rules, parameters the vendor supports — the vendor owns the compatibility of that behaviour across upgrades. When you customise — code that modifies or extends standard behaviour — you have signed a permanent contract with yourself. You now own that code's correctness, its documentation, its test coverage, and its re-validation every single time the platform moves underneath it. Multiply by the volume most estates accumulate and the annual upgrade stops being a routine event and becomes a project in its own right — which is why so many estates simply stop upgrading, and quietly fall out of support.

> Configuration is a setting the vendor keeps working. Customisation is a promise you make to maintain it yourself, forever.

The order of operations that keeps a core clean

The discipline is mostly a sequence, applied honestly to every gap between what the business wants and what the standard does.

BEFORE YOU WRITE A LINE OF CODE

- **Can the process change instead?** Often the requirement is a habit, not a necessity
- **Can standard configuration meet it?** The supported path the vendor keeps working
- **Can a clean extension meet it?** Built on documented interfaces, kept outside the core
- **Only then, custom code** — with an owner, a test, and a reason on record

The real cost is measured in upgrades not taken

The bill for an over-customised core is rarely paid at build time, when a customisation looks cheap. It is paid years later, as the accumulated weight of bespoke code makes each upgrade slower and riskier until the organisation decides the safest thing is not to upgrade at all. At that point the estate is frozen: still running, but drifting away from support, from security patches, and from every new capability the vendor ships. The customisations that were meant to make the business special have instead made it stuck.

Discipline, not abstinence

None of this argues for never customising. Genuine competitive differentiation sometimes justifies bespoke code, and refusing all of it is its own kind of failure. The discipline is simply to make each customisation a *decision* — one that names what it costs to carry, who will own it, and why the supported path wouldn't do — rather than a reflex. A clean core is not an empty one. It is one where every exception earned its place.

> **Where we sit.** Roltrader's PRAIS layer quantifies exactly this: what an estate's customisation actually costs to carry, and what staying current would take — computed from your own system, not a benchmark. But the discipline needs no tool. Ask the four questions in order, make every customisation a decision with an owner, and you keep the one thing that matters most: the ability to move.