GUIDE 24 June 2026 9 min read

ERP modernization without stopping the business

Why replacement projects stall, and the phased alternative that keeps operations running.

Amit Kumar
Founder, Udyat Technologies

THE SHORT VERSION

Do not replace your ERP. Surround it, move one process at a time, and let the old system shrink until switching it off is uneventful.

- Big-bang ERP replacement fails on sequencing far more often than on technology.
- Integration first: make the ERP one system among several, not the only door in.
- Move one process at a time, starting with whichever is costing the most today.
- Run old and new in parallel until the numbers reconcile, then retire properly.

Almost every business we talk to has the same story about their ERP. It was implemented years ago, it was customised heavily by someone who has since left, and it now sits at the centre of operations in a way that makes it both indispensable and impossible to change. Finance depends on it. Nobody wants to touch it.

The instinct is to replace it. A new platform, a clean implementation, a go-live date. That instinct is what turns an eighteen-month programme into a four-year one, and it is worth understanding why before committing budget to it.

The problem is almost never the technology. Modern ERP platforms are capable and the migration tooling is mature. What breaks these programmes is sequencing: the decision to change everything at once, in a business that cannot stop while you do it.

Why replacement projects stall

Four failure modes, in rough order of how often we see them cited as the reason a programme slipped.

A single cutover date

Everything depends on one weekend. Every slip in every workstream lands on that date, so the date moves, and each move costs a quarter. Nothing is deliverable until everything is.

Scope defined by the vendor

The module list becomes the plan. It reflects what the platform sells rather than what your operation actually struggles with, so the expensive work often lands somewhere that was never the bottleneck.

Customisations nobody owns

Years of small changes encode real business rules that exist nowhere else. Rediscovering them mid-migration is how a programme finds six months of unplanned work.

The business keeps moving

You are migrating to a target that changes underneath you. Eighteen months of new products, new regulations, and new customers all have to be built twice.

If a programme cannot deliver anything of value until the final cutover, it is not a modernization programme. It is a bet.

Signs the ERP is genuinely the constraint

Worth being honest here, because sometimes the ERP is fine and the problem is the twelve spreadsheets around it.

- Adding a product line, branch, or entity takes months and a consultant.
- Teams keep a parallel spreadsheet because the ERP screen is too slow to use live.
- The vendor's support for your version has ended, or the upgrade path is a reimplementation.
- Month-end close depends on one person exporting and reconciling by hand.
- You cannot answer a basic operational question without asking someone to run a report.
- Integration with anything modern requires a file drop and an overnight job.

Integrate before you replace

The first move is not migration. It is to stop the ERP being the only door into your data. That usually means putting a thin integration layer in front of it — an API surface, an event feed, or at minimum a reliable synchronised copy of the data other systems need.

This sounds like a detour. It is the opposite. Once other systems can read and write without going through the ERP's own screens, you can build new capability outside it without waiting for the migration. A supplier portal, a mobile capture app, an approval workflow — none of these need the ERP replaced first, and each one removes load from it.

It also does something quietly important: it turns the ERP into one system among several rather than the system. That is the state you need to be in before replacing it is safe, and many businesses discover that once they are there, the urgency to replace drops considerably.

Two ways to get to the same place

Both end with a modern platform. They differ in when you find out whether it is working.

Big-bang replacement

- One cutover date, everything moves at once
- No business value until the very end
- Risk concentrated into a single weekend
- Rollback means reverting the whole programme
- Every requirement must be known up front
- Typical horizon: 18–36 months

Phased, process by process

- One process moves at a time, each one live
- Value from the first phase onward
- Risk contained to the process being moved
- Rollback affects one process, not the business
- Later phases learn from earlier ones
- First phase live in 8–14 weeks

The sequence that works

This is the order we use. The principle behind it: every step has to be independently worth doing, so pausing after any of them still leaves the business better off.

01 — Map what the ERP actually does

Not the module list. The processes that genuinely run through it, who depends on each, and which customisations encode rules that exist nowhere else. This is where the unpleasant surprises are, and finding them now is far cheaper than finding them in month nine.

02 — Build the integration layer

A stable, documented way in and out. From here new systems can be built alongside the ERP rather than after it, and the ERP stops being a single point of change.

03 — Move the loudest process first

Whichever one is costing the most in manual effort or delay today — usually procurement, approvals, or dispatch. It is chosen for payback, not for how easy it is to migrate.

04 — Run both, reconcile, then switch

The new process runs alongside the old until the numbers match for a full cycle. Only then is the old path closed, and closed properly rather than left available as an escape hatch.

05 — Repeat, and let the ERP shrink

Each process that moves reuses the same integration layer and platform, so phase four costs a fraction of phase one. Eventually what remains in the ERP is small enough that replacing or retiring it is a routine decision.

What this costs, honestly

A phased programme is not automatically cheaper in total. What changes is when you spend and what you get for it. Instead of committing a large budget to a single outcome eighteen months away, you commit to a first phase that is live in roughly three months and either pays for itself or tells you something important.

The first phase is usually the most expensive per unit of work, because it includes the integration layer that every later phase reuses. That is the part worth funding properly. Businesses that try to save money by skipping it end up rebuilding a point-to-point integration for every subsequent process, and pay for it several times over.

The other honest point: this approach needs someone on your side who can make decisions about process, not just about software. If every question has to go through a committee, the phased model loses its main advantage, which is speed of learning.

What good looks like twelve months in

- ✓ Two or three processes fully off the old system and running better than before.
- ✓ Month-end close shorter, and not dependent on one person's spreadsheet.
- ✓ A clear, costed picture of what remains in the ERP and what it would take to finish.
- ✓ An integration layer that new work plugs into in days rather than months.
- ✓ Nobody re-typing the same order into two systems.
- ✓ The option to stop, without having wasted the money already spent.

The goal is not a new ERP. It is a business that is no longer constrained by the old one — and those are not the same project.