Tangle

ERP MIGRATION GUIDE

5 Pitfalls to Avoid

When Switching ERP Systems

Switching ERP is where good manufacturers lose months and goodwill. Most of the damage is self-inflicted, and every bit of it is avoidable.

Before you start

Replacing an ERP is one of the biggest operational changes a manufacturer will make. Done well, it clears out years of workarounds and gives the business room to grow. Done badly, it stalls production, frustrates the team, and costs far more than the license ever did.

Here is the part that surprises people: the failures rarely come from the software. Two systems that would both do the job perfectly well can lead to wildly different outcomes, depending entirely on how the switch is run. The risk lives in execution, not selection.

This guide is about that execution. It walks through the five mistakes that quietly sink ERP switches, why each one happens, and what to do instead. It finishes with a calmer, phased way to make the move and a checklist you can run against your own project.

First, have you actually outgrown your ERP?

Before worrying about how to switch, it is worth being honest about whether you need to. A few signs that you have outgrown your current system:

- You run the business from spreadsheets that sit beside the ERP, not inside it.
- Simple changes need the vendor, and come with a lead time and a quote.
- Nobody fully trusts the numbers without checking them by hand.
- New starters take months to become productive.
- You cannot see a job’s true status without asking two or three people.
- Reporting is a manual chore that eats the same hours every week.

If several of these ring true, the question is no longer whether to switch, but how to switch well. That is where the next five pages come in.

MISTAKE 01

Rebuilding your old mess in a new system

What it looks like

The team recreates the old system inside the new one, screen for screen, field for field, workarounds and all, because it feels familiar.

Why it happens

Familiarity feels safe, and change is uncomfortable in the middle of a project. Copying what exists is quicker than stopping to rethink it, so that is what tends to happen under time pressure.

What it costs you

You pay new-system money for old-system problems. The friction simply moves to a more expensive home, and you spend the one moment when everyone is paying attention to how work flows recreating the very things that slowed you down.

Warning signs

- Requirements are written as “make it work like the old system”.
- There are no current process maps, only tribal knowledge.
- The phrase “we’ve always done it this way” settles most decisions.

DO THIS INSTEAD **Map the process you want, then build that. Fix it on the way in.**

MISTAKE 02

Trying to migrate everything

What it looks like

The plan is to move ten years of history across, every customer, every part number, every closed job, just in case someone needs it one day.

Why it happens

It comes from a fear of losing something, plus the absence of a clear rule for what actually matters. With no rule, everything looks essential.

What it costs you

A slow, costly migration, and a brand-new system cluttered with dead customers and obsolete parts from day one. Reports get skewed by records nobody uses, and the clean start you paid for never really happens.

Warning signs

- No agreed data-retention rule.
- Nobody owns data quality, so nothing gets cleaned.
- The migration scope keeps quietly growing.

DO THIS INSTEAD **Migrate the data that runs the business today. Archive the rest where you can still reach it.**

MISTAKE 03

The big-bang go-live

What it looks like

Every department moves onto the new system on the same day, with the old one switched off behind them.

Why it happens

It looks simpler and cheaper than running two systems in parallel, and there is usually pressure to draw a line and be done with it.

What it costs you

Any single error, a mismapped field, a missing routing, a report that does not tie out, lands on a live production day with no way back. A small issue can stall the floor and burn the team's trust in the new system before it has had a chance to prove itself.

Warning signs

- No rollback plan if something goes wrong.
- No period where old and new run side by side.
- The go-live date is fixed before testing is finished.

DO THIS INSTEAD **Run in parallel, or phase the rollout by area, until the new system is proven on live work.**

MISTAKE 04

Buying the software but forgetting the people

What it looks like

The project is run by IT or the leadership team, and the people on the floor meet the new system for the first time at go-live.

Why it happens

It gets treated as a software purchase rather than a change in how people work day to day, so the human side is left until last.

What it costs you

Low adoption. People quietly keep their spreadsheets and side-processes, the ERP never becomes the single source of truth, and you end up paying for the system twice: once to buy it, and again in the workarounds that grow up around it.

Warning signs

- The shop floor had no say in the selection.
- Training is a single session near go-live.
- There is no internal champion in each area.

DO THIS INSTEAD **Involve the floor early, let them shape the workflows, train on real tasks, and name a champion in each area.**

MISTAKE 05

Choosing a system you'll outgrow

What it looks like

A system that dazzles in the demo but hard-codes today's requirements, so tomorrow's changes all become projects.

Why it happens

It comes from buying on feature lists and demo polish rather than on how easily the system can change once you own it.

What it costs you

Every change becomes a vendor ticket with a lead time and a quote attached. Two years in, you are back to spreadsheets for the things the ERP cannot flex to, which is exactly where you started.

Warning signs

- Small tweaks need professional services.
- Simple changes have long lead times.
- It takes heavy customization just to fit how you work.

DO THIS INSTEAD **Choose a composable, low-code ERP your own team can adapt as the business changes.**

A calmer way to switch

Avoiding the five pitfalls is easier if the whole project follows a sensible shape. This is the phased approach we see work again and again.

1. Get honest about the problem

Start with your real bottlenecks, not a feature wish list. Name the handful of things that cost you time and money today.

2. Model your real work

Give shortlisted vendors your own BOMs, routings or recipes and watch them run them live. Anyone who needs months to study requirements is already too slow.

3. Pilot on your hardest job

Test with your toughest planner and a real, awkward job. If the system cannot handle your hardest work, it will not handle the rest.

4. Migrate what matters, clean as you go

Bring across live customers, current parts, open and recent orders, and anything you must keep. Archive the rest and tidy the data on the way in.

5. Run in parallel, or phase by area

Prove each part of the system on real work before you depend on it. Start with quoting, then production, then purchasing, rather than flipping everything at once.

6. Go live, then keep improving

Treat go-live as the start, not the finish. Capture what people still do in spreadsheets and fold it back into the system over time.

Your migration checklist

A short, practical list you can run against your own project.

Before you start

- ☐ Write down the three or four problems the switch has to solve.
- ☐ Agree a data-retention rule (what moves, what gets archived).
- ☐ Map your core processes as they should work, not as they are.
- ☐ Name a champion in each area of the business.
- ☐ Set the measures you will judge success by.

During the project

- ☐ Pilot the system on your hardest job.
- ☐ Migrate live data only, and clean it as you go.
- ☐ Train people on real tasks, not generic demos.
- ☐ Run the new system in parallel, or phase it in by area.
- ☐ Keep a rollback plan until the new system is proven.

After go-live

- ☐ Measure against the success criteria you set at the start.
- ☐ Capture anything people still do in spreadsheets and address it.
- ☐ Keep adapting the system as the business changes.
- ☐ Retire the old system only once the new one is proven.

How Tangle de-risks the switch

Tangle is built to make the move less risky, not more dramatic. A few things that help directly with the pitfalls above:

- Model your work in hours, not months, using pre-built templates for discrete, batch and engineer-to-order manufacturing.
- Composable and low-code, so you turn on only what you need and change it yourself, without a vendor ticket for every tweak.
- Run in parallel with your existing system while Milo, Tangle's native AI engineer, helps you shape the ERP in a safe sandbox before anyone switches over.
- Bring across the history that matters and map it to the right places, so you start clean rather than cluttered.

The thread running through all five

Notice what these five have in common. None of them is really about the software. They are about scope, data discipline, sequencing, people and flexibility. Get those right and almost any capable system will serve you well. Get them wrong and the best software on the market will still disappoint.

If you are weighing up a switch, that is also the fastest way to build a shortlist. Ask each vendor to prove they help you avoid these five, using your parts, your data and your team, not a canned demo.