

WHY YOUR ERP PROJECT IS ALREADY BEHIND

A playbook from Trail Guide Consulting Group, for leaders running a manufacturing plant through an implementation.

Before we start: your vendor is not on your side of the table

That is not an accusation. It is just the structure. The implementation partner is paid to deliver the software they sell. You are the one who has to ship product after go-live with whatever that software turns into. Most of the time those two jobs overlap. Somewhere around data migration or preparing for pilot, they stop overlapping, and the person who finds out first is you.

I work independently. I have no stake in what you buy or how it gets configured. That matters because almost every other voice in the room has a financial reason to prefer one answer. My job is to sit on your side of the table during the implementation, keep the vendor accountable, and help your team make decisions and produce their deliverables in a manner that protects your timeline and your budget.

I came to this from the plant side. I have over 25 years of engineering, finance, and operations experience. I have led ERP deployments on platforms like Infor Visual Manufacturing, Acumatica, and Dynamics AX, and I have real shop-floor experience, not just software knowledge. When I walk your floor and talk to the people doing the work, I am not learning manufacturing for the first time.

Projects rarely fall behind in one dramatic moment. They lose a few hours here and a few days there, in places nobody is watching, until the date quietly stops being real. These are the five places I see it happen most.

01 Your team is doing the project on top of their day jobs

The people who know your data, your routings, and your workarounds are the same people who keep the plant running. When the implementation needs them, the plant does not pause. So the project gets whatever hours are left over, and the schedule absorbs the difference without anyone saying so.

Nobody is bad at either job. They just cannot do both. The warning signs show up early: test sessions rescheduled twice, questions that sit for a week because the one person with the answer is on the floor, and decisions made by whoever happened to be in the room.

The fix is unglamorous. Name the people the project depends on, decide how many hours a week each one is actually released from their regular work, and write it down. Then either backfill those hours or accept that the date moves. If the plan assumed twenty hours a week from your lead planner and the real number is six, you already know where the schedule is going.

02 “That’s a customization” or “that’s a change order” becomes the answer to everything

Every implementation hits a gap between how the software works out of the box and how your plant actually runs. The vendor’s answer is often a customization or a change order, and each one sounds small and reasonable on its own. A custom report here, a modified screen there, a special routine for one customer.

Individually, they are fine. Together, they are the reason the budget and the timeline stop meaning anything. Each one has to be specified, built, tested, and documented. Customizations rarely take the initial hours estimated. Someone on your side pays for that long after go-live because they have to be carried forward through every future upgrade.

Before you approve the next one, ask two questions. Is this how we want to run the plant, or just how we have always run it? If we changed the process instead of the software, what would it cost us? Sometimes the customization is the right answer. But it should be a decision you make on purpose, as a last resort, and with a price and an owner.

03

The legacy data has no owner and no plan

Everyone agrees the data needs cleaning. Nobody is responsible for it. Often it moves into the new system exactly as broken as it was in the old one, except now it is live, and every transaction touches it.

Inventory counts, bad item masters, bills of material that do not match what the floor builds, routings nobody has updated in years, and customer and vendor records with three versions of the same name... none of this is a software problem. It is an ownership problem, and software vendors won't own it, because it is not in their scope.

Assign a named person on your side to own each data set: items, BOMs, routings, customers, vendors, open orders, and inventory balances. Give them a deadline that sits well ahead of the pilot date, not on it. Plan more than one trial load, because the first one will find problems you did not know you had. I wrote more about this in ["Garbage In, Garbage Live."](#) and the checklist later in this guide is where to start.

04

The configuration matches the demo, not your floor

The system gets set up around how the software expects work to flow. Your shop has years of real reasons why it flows differently. Only one of those two is going to bend, and if you are not watching, it will be your floor.

FIELD STORY

I have seen this firsthand. At my second go-live, I championed a new labor collection process that passed every test we ran. On the first day of live production, it failed. The old system had let operators share a generic login at a workstation. The new one tied each login to one person, so when one employee signed in, it kicked a coworker off. Nobody had tested for that because nobody in the test room worked the way the floor worked. We had to roll the process back that same day. It wasn't my finest day.

The lesson is not that testing is useless. It is that a test script written from the software's point of view will pass on the software's terms. Put the people who do the work in front of the system, running a real job the way they run it, before you sign off on a configuration. The map is not the trail, and the people who walk the trail every day are the ones who will notice what the map left out.

05

Nobody in the room is paid to tell you the date will slip

The vendor wants the milestone signed. Your team does not want to bear the bad news or advertise the project isn't going well. The project manager is watching a status report that has been green for weeks. The first honest conversation about the schedule happens after the schedule has already slipped.

This is the one that ties the other four together. Each of them creates slippage, and none of them shows up on a status report until it is too late to fix cheaply.

You need at least one source of truth that does not depend on anyone's incentive to report good news. That can be a short weekly review where you ask for the work remaining, not the work completed. It can be an independent set of eyes on the plan. What matters is that someone is asking, early and regularly, whether the date is real, and that the answer is allowed to be no.

06

Pre-go-live data readiness checklist

Use this a few months before your go-live date, and again before each trial load. If you cannot check a box, you have found a risk while it is still cheap to fix.

- ☐ Every data set (items, BOMs, routings, customers, vendors, open orders, inventory) has a named owner on our side, not just the vendor's
- ☐ Each data owner has a deadline that falls before pilot, not on it
- ☐ Obsolete and duplicate item numbers, customers, and vendors have been identified and either cleaned or deliberately excluded
- ☐ Bills of material and routings have been walked against what the floor actually builds, by someone who works on the floor
- ☐ Units of measure, lead times, and costs have been spot-checked on a sample of active items
- ☐ At least one full trial static data load has been run, and the errors it found are logged with owners and dates
- ☐ Inventory counts are scheduled to be reconciled close to cutover by a physical count, with a plan for transactions made in between
- ☐ Open orders and open purchase orders have been cleaned up and there is a migration plan detailing how they move over and who verifies them afterwards
- ☐ Someone outside the project team has reviewed the results of the last trial load and signed off in writing

07

Questions to ask your vendor before you sign the next change order

A change order is a decision point. Ask these before you sign, and write down the answers.

- 01 What exactly is changing in scope, and what is the effect on the go-live date? A number of days, not "minimal."
- 02 Is this a gap in the software, or something that was in the original scope and was missed?
- 03 Is there a way to meet this need with standard functionality and a process change? What would that cost us?
- 04 Who builds it, who tests it, and who on our side has to be involved? For how many hours?
- 05 How does this customization behave through future upgrades, and who takes the P&L hit to maintain it?
- 06 If the change slips, what happens to the rest of the schedule?
- 07 What is the total cost to date against the original budget, and what is the forecast to finish?

CHECK THE MAP AGAINST THE TRAIL

If you read the five places above and recognized your own project in even one of them, you are not alone, and you are not too late. Most of these problems are cheap to fix when they are caught early and expensive when they are found at go-live.

The vendor's plan is the map. Your plant is the trail. My job is to walk both, tell you where they disagree, and make sure the decisions that follow are yours, made with the full picture.

If you would like a second set of eyes on where your project stands, schedule a free [30-minute call](#).

trailguide-consulting.com
sales@trailguide-consulting.com
(458) 200-4191