RIGHTTACK CONSULTING · PRODUCT DEFINITION & PLM

How to spend three million dollars and still have your engineers emailing spreadsheet BOMs

PLM is a discipline, not a licence. A field guide to the product record — why the definition has to exist before the tool, and how a system nobody trusts ends up in a background tab while the real bill of materials lives in a file called FINAL_v3.

Open the shared drive of almost any hardware company that grew faster than its processes and you will find it: a folder, or a file, called something like LATEST_REV_FINAL_v3_USE_THIS_ONE. That file is the real bill of materials — not the one in any system, but the one the engineers actually email each other, because it is the only one they trust. Procurement orders to a revision engineering superseded a fortnight ago. The unit in the field does not match the drawing that supposedly defines it. And nobody can answer the simplest question in manufacturing — what exactly are we shipping today — without walking down the hall to ask a particular person. This is the problem PLM is sold to fix. It is also the problem a badly bought PLM quietly makes worse.

Here is how that second part goes, and it is common enough to have a shape. A company feels the pain, runs a selection, and spends real money — call it three million dollars, all in — on an enterprise PLM. Executive sponsorship. A multi-year project. A celebrated go-live. And eighteen months later the engineers are back on the spreadsheet, the PLM is open in a background tab nobody looks at, and the folder is now called FINAL_v4. The software did not fail. Almost none of them do. What failed was everything around it.

PLM is a discipline, not a licence

The first mistake is in the category itself. People hear “PLM” and picture software — a thing you buy, install, and are then done with. But Product Lifecycle Management is not software. It is the discipline of holding one authoritative record of what your product is, from concept through design and manufacture to service and end of life. The software enables that discipline; it does not create it. Buy the licence without building the discipline and all you have done is give your existing chaos a database to live in.

> **A PLM licence is not a defined product. It is a place to keep one — if you have one.**

Know what each box is actually for

The second mistake is drawn before a vendor is ever met: not knowing where PLM ends and the systems on either side of it begin. PDM sits on the engineering side — CAD file management, document versioning, BOM authoring — and stops at the engineering team's boundary. ERP sits on the far side; it knows the product exists and what it costs, but not how it was designed or why. PLM is the span between them: the full product record, the change process, the compliance trail, and the single master BOM that every other BOM — engineering, manufacturing, service — is derived from. It carries **what you make**. ERP carries what you hold and what you owe. Draw that boundary yourself, on paper, before the demos start, because every vendor will happily draw it for you in the way that sells one more module.

What it buys when it works

It is worth being concrete about the prize, because the failures are loud and the wins are quiet. A working product record gives you one authoritative answer to "which revision is current" — not six, scattered across inboxes and shared drives. Engineering change runs as a controlled process with cross-functional sign-off and a full audit trail, rather than a change travelling by email and verbal say-so. One master BOM feeds the engineering, manufacturing and service views, so they stop quietly drifting apart. Suppliers work from the approved revision, on time, seeing only what they need — which is usually where the fastest visible payback sits: fewer wrong-revision builds, shorter corrective-action cycles. And compliance — RoHS, REACH, conflict minerals — is tracked at the component level, continuously, instead of being reconstructed in a panic the month before launch. None of that is glamorous. All of it is the difference between a product you control and one that controls you.

Death by spreadsheet

Go back to the three-million-dollar failure, because the autopsy is always the same five findings. The system made the job harder — more clicks, more approvals, slower access than the shared drive it replaced, and a tool that slows the daily work gets routed around every single time. The data was never trustworthy — BOMs migrated with gaps, part numbers duplicated, documents linked to the wrong revision — so the first time an engineer opened a record they knew was wrong, they quietly stopped believing the system. Training was a one-time event at go-live, so every new hire learned the spreadsheet workaround from a colleague rather than the process from the tool. Workflows were configured for the ideal process, not for how engineering actually works under schedule

pressure; when the two collided, speed won. And nobody owned it after launch — the implementer left, IT closed the ticket, no one cleaned the data or updated the workflows as the product line evolved, and entropy did the rest inside a year.

> **The software almost always works. The implementation, the change management and the governance almost always do not.**

What PLM will not do for you

Which points straight at the section the vendors skip. PLM automates and enforces your process — so if that process is undefined or dysfunctional, PLM enforces the dysfunction, faster and at greater cost. It is only ever as good as the data you put in it and the discipline you run it with. It will not fix a broken product-development process. It will not make a supplier hit a date. It will not write your product strategy. And it will not supply a definition you never built. Garbage in, garbage out is not a throwaway line here; it is the single most reliable predictor of whether the three million dollars buys a competitive advantage or a background tab.

Two of those limits are organisational, and between them they kill more implementations than any feature gap. PLM touches every team that touches the product, so it needs an executive who owns the outcome and can break the logjams; without one, adoption stalls and the thing curdles into “engineering’s system.” And it is not an IT project. IT owns the infrastructure; the business owns the implementation. When IT leads and the business rides along, the tool gets built for IT rather than for the people who have to live in it — which is another way of describing a background tab.

So the real first step is not selection. It is checking that the thing you are about to systematise exists at all: part numbers that are unique and mean one thing, a rule for when a change takes a new revision and when it takes a new part number, a bill of materials that drives the build rather than sitting beside it as a second opinion. Lay a PLM over that skeleton and it earns every dollar. Lay one over its absence and you have paid to reproduce the gap — faster now, and behind a login that makes it look authoritative.

If you are going to buy one, buy it like this

None of which is an argument against PLM. A defined product record is a genuine competitive advantage, and past a certain complexity you cannot run the discipline on a spreadsheet no matter how good your people are. It is an argument for buying, and standing one up, in the right order.

> THE ORDER THAT SURVIVES CONTACT WITH A SCHEDULE
>
> - **Requirements before demos.** Write them down, weight them by business criticality, and score against them. An unmet must-have disqualifies a vendor; a missing nice-to-have is a rounding error.
> - **A three-year cost of ownership.** The licence is rarely the largest number — implementation, integration, training and the headcount to run it usually are.
> - **Phase it.** Define ownership and clean the data before anything moves; pilot on one product line; hard cutover with real hypercare; expand to the next module only once the last is stable.
> - **Configure, do not customise.** Every customisation is debt you repay on every upgrade.
> - **Name an owner on day one.** A business role, not an IT one, who owns data quality, adoption and the change process — permanently.

The mechanics rhyme with any serious system decision. The difference with PLM is that the thing being systematised is the definition of your product, so the cost of getting the data wrong does not stay put — it compounds through every build, every change and every field failure that follows.

PLM done right is a competitive advantage. Done wrong, it is an expensive lesson with your logo on it. And the line between the two is not the software — it is whether the discipline existed before the licence did, and whether anyone owns it after the consultants leave. The tool can hold your product record. It cannot decide that you will keep one.

None of this is a purchase you make once. It is a definition you have to build and then defend — part numbers, a revision rule, a change process, a single BOM — with a system stood up on top of it in the right order and owned after go-live. Knowing the discipline comes first is the easy half. Building it, and holding it against a schedule that rewards the spreadsheet, is the work.

That is the work I do.

WHERE THIS GOES NEXT

Knowing exactly what you ship →

The product definition and change control method — the skeleton a PLM is meant to carry and cannot create. Or start a conversation: rtackconsulting.com/contact