

Just because SAP can, doesn't mean you should

How experienced teams decide what not to
implement in SAP EWM



Introduction

SAP Extended Warehouse Management (EWM) is a powerful platform. It offers deep functionality across inbound, storage, processing, picking, packing, shipping, labour, and automation integration. For many organisations, the risk is not that SAP EWM *cannot* support their warehouse, it is that it can support **far more than the operation is ready to absorb**.

This whitepaper explores a rarely discussed aspect of warehouse delivery: **knowing when *not* to implement a feature**.

Drawing on real design experience, it explains why restraint is often a stronger indicator of expertise than breadth of configuration and how saying “not now” can protect operations, reduce risk, and create space for sustainable evolution.

The natural temptation to use everything

SAP EWM implementations often begin with enthusiasm:

A clean slate

A long wish-list

A powerful system

Stakeholders keen to “get everything we’re paying for”

Features are reviewed, demonstrations are convincing, and decisions are made early, often before operational readiness is fully understood.

The unintended consequence is that warehouses can go live with:

Unused functionality

Over-complex flows

Operator confusion

Capability that exists technically but not practically

The system works. The warehouse struggles. Capability is not the same as readiness. A core principle of experienced EWM delivery is simple: Capability answers “can the system do this?” Readiness answers “should we do this now?”

Warehouses are socio-technical environments:

People

Processes

Data

Physical constraints

Time pressure

“Implementing functionality that the system supports, but the operation cannot yet sustain, shifts risk from IT to the warehouse floor. That risk rarely appears in blueprints. It appears at 06:00 on a Monday.”



Andy Taberner,
Senior Solutions Architect, TCT

The hidden cost of over-design

Over-design does not always look like failure. More often it looks like:

- Manual overrides becoming routine
- Processes being “temporarily” simplified
- Certain system options being quietly ignored
- Experienced operators bypassing intended flows
- Support teams normalising exceptions

None of this shows up as a go-live defect yet all of it erodes system value. In many cases, a smaller, more constrained initial design would have delivered more real benefit.

Where experienced teams commonly pause

In hands-on EWM design, there are several areas where experienced teams frequently choose *not* to implement full functionality at first even though SAP EWM supports it well. These decisions are not anti-SAP; they are pro-operation.

Handling Unit Management

Handling Unit (HU) management is one of SAP EWM's strengths, but it requires:

- Consistent inbound quantities
- Disciplined labelling
- Stable master data
- Time at receipt

In environments where:

- Inbound arrives in small, mixed quantities
- Palletisation happens dynamically
- Speed matters more than traceability
- Full HU management can slow inbound and increase friction.

Experienced designs may:

- Use HUs only outbound
- Allow HUs but not require them
- Defer full HU management until processes stabilise

This is not avoidance, it is sequencing.

Interleaving and labour optimisation

Interleaving promises efficiency by combining tasks and reducing empty travel. In practice, it relies on:

Accurate, stable bin coordinates

Predictable task volumes

Full pallet movements

Highly structured flow

Where these conditions are not met, interleaving creates:

Assignment confusion

Missed priorities

Increased supervisory intervention

“Choosing *not* to enable interleaving initially often protects flow and operator confidence.”



Damon Conway,
Senior Solutions Architect, TCT

Deconsolidation and advanced work centres

Deconsolidation, kitting, and advanced work centres are powerful but they require:

Space

Clear routing logic

Dedicated resource time

Handling unit discipline

In warehouses without physical or organisational capacity to support them, enabling these processes in the system simply displaces the problem. Restraint prevents the system being blamed for constraints it cannot fix.

Inbound quality and inspection processes

Quality inspection functionality assumes:

- Available inspection space
- Defined inspection outcomes
- Clear disposition paths
- Capacity to interrupt flow

Where warehouses are inbound-volume constrained, inspecting everything may be impossible.

Experienced teams will:

- Design for status change rather than physical inspection
- Apply inspection selectively
- Defer full QM-driven inbound until the operation can support it

“Not now” is a design decision, not a compromise . The most important distinction is this: Saying “*not now*” is different from saying “*never*”.



Phil Foulger,
Senior Solutions Architect, TCT

Good designs:

- Keep doors open for future capability
- Avoid configuration dead ends
- Use flexible settings where possible
- Document intent clearly

This allows evolution without re-platforming, and ensures that future enhancements are additive rather than corrective.

Why large programmes often struggle with restraint

Large transformation programmes can unintentionally encourage over-design:

Templates make everything look achievable

Standard demos imply low effort

Governance discourages deviation once decided

Delivery teams may not own post-go-live outcomes

In this environment, the safest path can appear to be “implement it now.” Specialist, delivery-led teams tend to behave differently:

They expect to support what they design

They see how features behave under pressure

They are comfortable pushing decisions back when readiness is unclear

Restraint requires confidence.

Restraint protects operators and system credibility

Warehouses form opinions about systems very quickly. If the system:

Slows people down

Asks for data they don't have

Offers options they can't realistically use

Trust erodes.

“By limiting initial scope to what genuinely supports execution, the system earns credibility, making future enhancements easier to introduce and adopt”.



Blaine Tracey,
Senior Functional Consultant, TCT

Restraint is also an AMS strategy Every advanced feature introduced:

- Adds configuration surface area
- Increases support scenarios
- Creates more exception handling
- Requires onboarding and training

Implementing only what is operationally required at go-live:

- Reduces early support noise
- Improves stability
- Allows AMS teams to learn the environment properly
- Creates a controlled baseline for future change

Less now often means more later.

How experienced teams decide to hold back

Restraint is not instinctive it is structured. Good teams routinely ask:

- What problem does this feature solve today?
- What happens operationally if we *don't* implement it?
- Where does the risk sit, system or warehouse?
- Does the operation have the discipline to sustain it?
- Will this reduce or increase dependency on workarounds?

If the answers are inconclusive, the safest decision is often to wait.

Real maturity is knowing your warehouse, not your system

Organisations rarely regret not switching on a feature at go-live. They often regret:

- Introducing complexity too early
- Expecting behaviours the operation could not sustain
- Designing ambition into places where capacity was fixed

Expert EWM delivery is not about maximising system usage, it is about maximising operational benefit at the right time.

Conclusion: experience shows in the features you delay

Anyone can demonstrate SAP EWM capability. Experience shows in:

What you deliberately defer

What you protect operators from

What you sequence rather than rush

What you leave optional instead of mandatory

Restraint is not conservatism. It is respect for warehouse reality. Just because SAP EWM *can* do something does not mean your warehouse *should*, yet.

©TCT



Get in touch today
theconfigteam.com