

CASE 01

Multi-Channel Inventory & Operations Control

From fragmented stock signals to explicit business rules and reliable exception handling.

Decision question

How should an e-commerce operator redesign inventory ownership, allocation and exception handling across WooCommerce, Airtable, warehouse stock and dropship suppliers without creating a risky full-platform rebuild?

OPERATING MODEL Own stock + dropship	CORE SYSTEMS 5 connected tools	PRIMARY RISK Oversell / misrouting
ANALYSIS SCOPE Order-to-fulfilment	DECISION HORIZON 8-12 weeks	OUTPUT Target workflow + requirements

ANALYTICAL FRAMEWORK

1 Question Define the decision	2 Evidence Structure inputs	3 Analysis Test issues	4 Priorities Rank exposure	5 Options Support action
---	--	---	---	---

Illustrative demonstration case

This case is a fictionalised scenario created to demonstrate analytical method, decision structure and deliverable design. It does not represent a named client engagement.

01 / ENGAGEMENT FRAME

Business context and problem definition

The operating issue is not a single automation failure. It is a control-design problem: several systems hold overlapping information, while ownership and priority rules are partly implicit.

Illustrative current state • WooCommerce captures customer orders and displays sellable stock. • Airtable is used for operational coordination, supplier status and manual exceptions. • The warehouse holds physical inventory and reports availability separately. • Three dropship suppliers provide stock through feeds with different refresh intervals. • Slack and Intercom are used to resolve order exceptions and customer questions.	Symptoms observed • Occasional overselling when supplier stock changes between refresh cycles. • Orders can be manually re-routed without a consistent audit trail. • Customer support cannot always see the true fulfilment source or delay reason. • Returns and cancellations update some systems faster than others. • Teams rely on individual knowledge for exception handling.
--	--

DECISION CRITERIA

What the solution must achieve

Criterion	Required outcome	Failure mode to avoid
Inventory authority	One authoritative owner per stock signal	Competing systems overwrite each other
Allocation	Explicit priority rules by SKU / source	Ad hoc fulfilment decisions
Exceptions	Named exception states and owners	Issues disappear into Slack
Auditability	Reconstruct order decisions end-to-end	No evidence of why routing changed
Customer visibility	Support sees status and next action	Repeated internal escalation
Implementation risk	Incremental change with rollback path	Big-bang rebuild disrupts orders

Core analytical question Which system should own each data object, which business rules determine order allocation, and which exceptions require human intervention?
--

Portfolio note: all company names, figures and operating details in this demonstration case are illustrative.

02 / DIAGNOSTIC

Current-state control analysis

The diagnostic separates data ownership from workflow orchestration. A system can coordinate a decision without being the authoritative source for the underlying data.

Data object	Observed owner	Control issue	Target authority
SKU / product master	Airtable + WooCommerce	Duplicate edits and naming drift	Single product master
Own-stock availability	Warehouse + WooCommerce	Timing mismatch	Warehouse / WMS feed
Dropship availability	Supplier feeds	Different update intervals	Supplier feed by source
Customer order	WooCommerce	Stable, but downstream status fragmented	WooCommerce transaction record
Allocation decision	Manual / Airtable	Rules partly implicit	Central rule engine / orchestration layer
Exception status	Slack + Airtable	Unstructured ownership	Structured exception register

KEY FINDINGS

What is actually driving the failures

- 01 The stack lacks an explicit source-of-truth model. Several tools are being asked to answer the same question with different update speeds.
- 02 Allocation logic exists as operational habit rather than formal business rules. This creates inconsistent decisions when the preferred source is unavailable.
- 03 The exception process is a communications process, not a controlled workflow. Slack messages can trigger action but do not provide durable state, ownership or evidence.
- 04 Inventory freshness is treated as binary. In practice, supplier stock should carry a confidence or staleness threshold before it is exposed as sellable.
- 05 Returns, cancellations and partial fulfilment are not edge cases; they are state transitions that must be designed into the data model.

Control principle

Separate authoritative data, orchestration logic and human exception handling. Do not solve all three by making one tool the universal system of record.

Portfolio note: all company names, figures and operating details in this demonstration case are illustrative.

03 / TARGET DESIGN

Recommended operating model

The target state preserves the existing stack where possible, but makes ownership, routing and exception rules explicit.

A Order capture WooCommerce	B Availability Source feeds	C Allocation Rule engine	D Fulfilment Warehouse / supplier	E Exception Owned workflow
---	---	--	---	--

Rule area	Example business rule	Operational effect
Source priority	Use own warehouse first when ATP > safety stock; otherwise approved dropship source by rank	Predictable routing
Freshness	Do not promise supplier stock if feed is older than defined threshold	Reduces stale-stock oversell
Split orders	Only split when margin and service thresholds are met	Controls fulfilment cost
Manual override	Require reason code, owner and timestamp	Creates audit trail
Cancellation	Release reservation before customer status is closed	Prevents phantom stock
Returns	Route returned stock through inspection state before available-to-promise	Avoids premature resale

EXCEPTION TAXONOMY

Human intervention is reserved for defined states

Operational exceptions - No valid fulfilment source - Supplier confirmation timeout - Price or margin threshold breached - Partial fulfilment mismatch	Control exceptions - Conflicting stock values - Unmapped SKU / supplier code - Manual override requested - Return state inconsistent across systems
---	--

Design choice Airtable remains useful as an orchestration and operations layer, but should not overwrite source-owned inventory facts simply to create a single-looking database.

Portfolio note: all company names, figures and operating details in this demonstration case are illustrative.

04 / OPTIONS

Implementation choices and trade-offs

Three viable routes exist. The preferred route balances control improvement with implementation risk and preserves the option to migrate later.

Option	Scope	Advantages	Exposure	Assessment
A - Stabilise current stack	Document rules; tighten existing automations	Fastest, lowest cost	Technical debt remains; limited observability	Defensive
B - Add orchestration layer	Formal rules, structured exceptions, monitored integrations	Strong control uplift without full rebuild	Requires disciplined data contracts	Preferred
C - Full OMS / platform migration	Replace orchestration and fulfilment architecture	Highest long-term standardisation	Cost, migration risk, operational disruption	Only if scale justifies

RECOMMENDED ROUTE

Option B - controlled orchestration layer

- Create a canonical data dictionary and source-of-truth matrix before changing automations.
- Define allocation rules in human-readable business logic, then translate them into developer-ready requirements.
- Use structured exception states with named owners, SLA timers and reason codes.
- Instrument every integration with timestamps, error states and replay / reprocessing capability.
- Run parallel validation on a limited SKU / supplier set before expanding scope.

Why this route

It improves control and auditability while preserving optionality. If future volume justifies an OMS migration, the business rules and data contracts already exist as migration assets.

Portfolio note: all company names, figures and operating details in this demonstration case are illustrative.

Execution roadmap and example outputs

The work is sequenced to reduce irreversible change. Each phase produces a usable artefact before code or automation changes are committed.

Phase	Primary work	Deliverable	Gate
1. Map	Systems, data objects, workflow, failure points	Current-state map + data dictionary	Ownership agreed
2. Rules	Allocation, reservation, cancellation, returns	Business-rule catalogue	Operations sign-off
3. Design	Target flow, exceptions, integration contracts	Target-state blueprint	Developer review
4. Build	Automation / integration changes	Configured workflow + monitoring	Controlled test
5. Validate	Edge cases, replay, rollback, user acceptance	Test pack + defect log	Go / no-go
6. Operate	KPIs, incident ownership, runbook	Control dashboard + SOP	Handover

EXAMPLE DELIVERABLES

What an engagement can produce

Decision & control artefacts • Source-of-truth matrix • Business-rule catalogue • Exception taxonomy • RACI / ownership map • Risk and dependency register	Implementation artefacts • Developer-ready requirements • Integration data contracts • Edge-case test scenarios • Rollback / reprocessing logic • Operations runbook and KPI definitions
---	---

What this case demonstrates

Business process analysis, systems thinking, operational risk control, developer coordination and translation of ambiguous workflows into testable rules.

Portfolio note: all company names, figures and operating details in this demonstration case are illustrative.