Modelling an Enterprise Architecture Landscape as a Value Network

A graph-theoretic approach to enterprise integration analysis

Working Document — April 2026

Authored by John Siegrist <john@complects.com>

1. Purpose

This document describes a method for representing the enterprise application landscape as a **value network** — a graph whose nodes are business systems and whose edges are named value exchanges between them. The goal is to produce a model that is:

Strategically readable by non-technical stakeholders.

Mathematically tractable for graph-theoretic analysis.

Seamlessly decomposable into finer-grained views without losing the business meaning.

The approach separates what business value flows between systems from how it is technically transported. No protocols, message brokers, or file transfer mechanisms appear as edges. Domain-aligned data stores that expose business-facing services (e.g., a Customer Data Platform, a Product Master) are modelled as first-class system nodes.

2. Core Concepts

Concept	Definition	Example
System Node	A deployable or logical application that refines one or more business capabilities. Internally, it may contain its own component architecture — the landscape model treats it as a single unit.	"Order Management System" refines Order Capture, Order Orchestration, Payment Integration, Fulfilment Status Hub.
Capability	A what the business does, independent of the system that implements it. Shown inside a system node to indicate what that system is responsible for.	"Inventory Management" is a capability refined by the ERP system.
Value Exchange	A named flow of business information or intent between two systems. Describes the business meaning of the connection — never the transport mechanism.	"Order Submit", "Unified Customer Profile", "Shipment Confirmation".
Altitude	The level of abstraction at which a value exchange is described. Following Cockburn's use case altitude model: kite-level exchanges summarise a strategic integration purpose; sea-level exchanges describe specific, complete business interactions.	Kite-level: "Order Management". Sea-level: "Order Submit", "Order Modification", "Order Cancellation".

A kite-level value exchange groups the sea-level exchanges that serve a single strategic business goal between two systems. This grouping is not arbitrary — the formation rule is shared business outcome. If

two sea-level exchanges serve different business goals, they belong under different kite-level summaries.

3. Two Views at Different Altitudes

The value network can be read at two altitudes, with a clean symmetry between node refinement and edge refinement:

Kite-Level View — System nodes connected by kite-level value exchanges. Each edge summarises a strategic integration purpose. This is the view for executive and programme-level conversations.

Sea-Level View — The same system nodes (with their internal capabilities visible) connected by sea-level value exchanges — each a specific, complete business interaction. This is the view for architecture and delivery conversations.

Both views describe the same network. The kite-level view is a summary of the sea-level view, not a separate model. Architects can move between altitudes without changing the underlying topology.

Each system node in the landscape is a boundary — a point where two systems' internal architectures meet. A value exchange between two systems implies an interface on both sides: one system produces or sends, the other receives and acts. A fuller decomposition of what happens inside each system is possible but outside this document's scope.

4. Example: Retail Company Landscape

A simulated small enterprise is used for illustration. The business context is a retailer moving from a fragmented, batch-oriented legacy landscape to a real-time, event-driven future state.

4.1 Current State (Kite-Level)

The as-is landscape is characterised by batch integrations, manual handoffs, and an ERP-centric architecture where most value flows route through SAP.

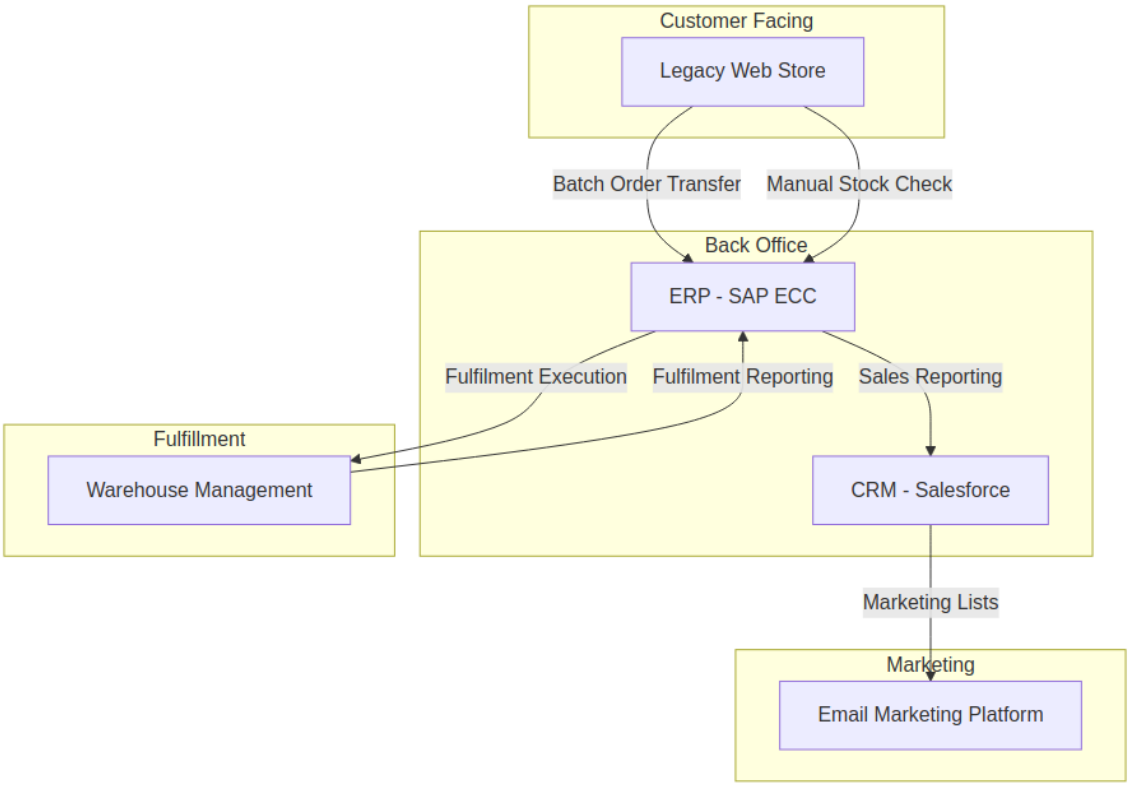

Fig. 1 — Current-state landscape at kite level. Note the ERP as a bottleneck through which nearly all value flows are routed.

4.2 Future State (Kite-Level)

The target architecture introduces an Order Management System and Customer Data Platform, redistributing integration load and enabling real-time value flows.

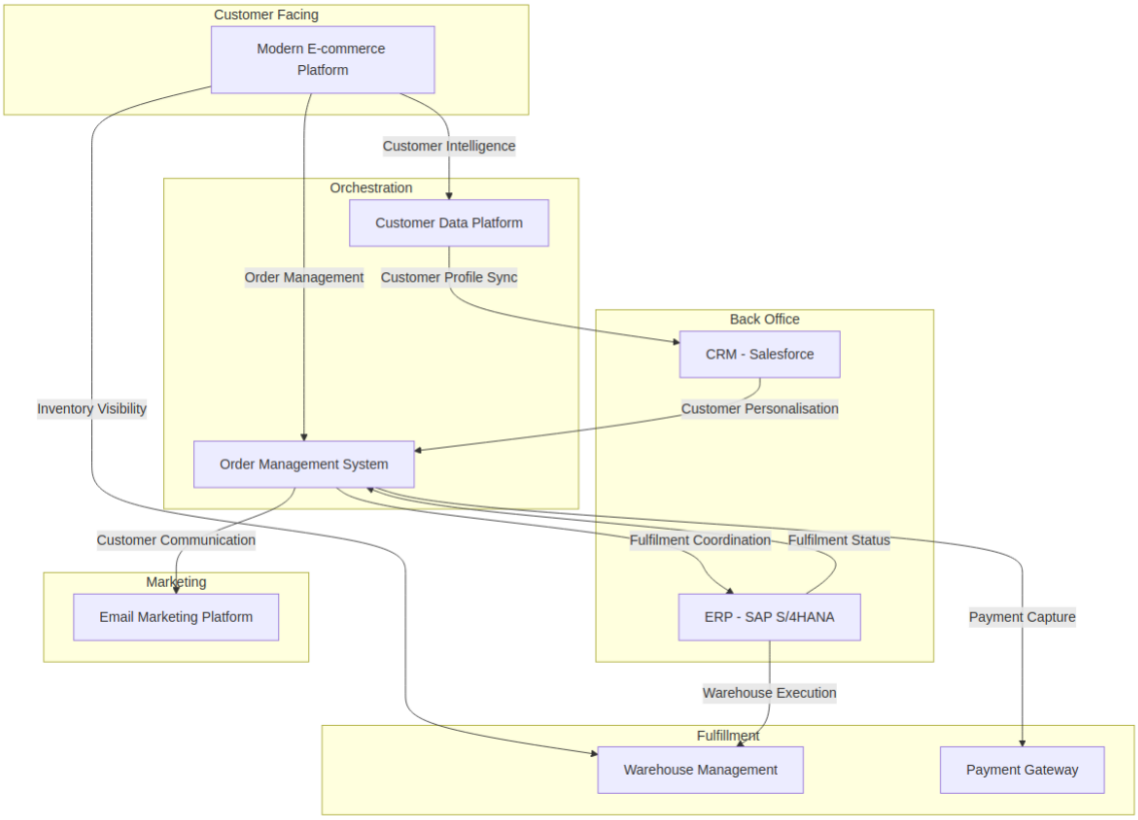

Fig. 2 — Future-state landscape at kite level. Integration load is distributed across OMS and CDP rather than concentrated in the ERP.

4.3 Future State (Sea-Level)

The sea-level view refines both nodes (showing capabilities inside each system) and edges (showing the specific value exchanges that realise each kite-level summary).

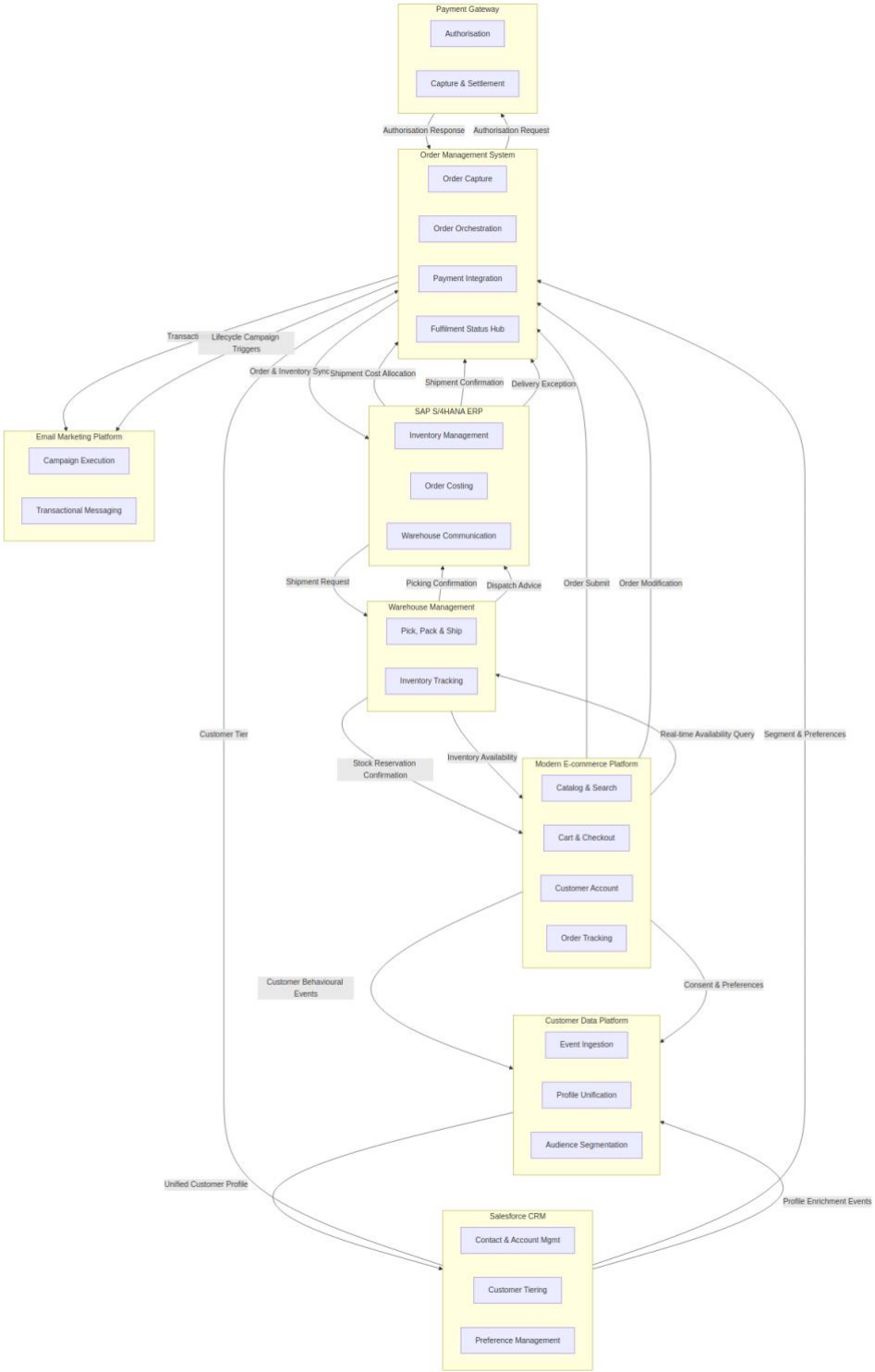

Fig. 3 — Future-state landscape at sea level, showing capabilities per system and individual value exchanges.

The mapping between altitudes is straightforward: each kite-level edge (Fig. 2) is realised by a group of sea-level edges (Fig. 3). For example, the kite-level exchange “Order Management” between eShop and OMS is realised by the sea-level exchanges “Order Submit” and “Order Modification”. This mapping can be stored as an edge attribute.

5. Current State vs. Future State: A Worked Comparison

Placing the two kite-level views side by side makes the transformation visible as a set of graph operations.

Change Type	Specific Changes
Nodes added	OMS, CDP, Payment Gateway, Modern E-commerce Platform
Nodes removed	Legacy Web Store
Nodes evolved	ERP (SAP ECC → S/4HANA), CRM and WMS retained
Edges added	10 new kite-level value exchanges (Order Management, Customer Intelligence, Payment Capture, etc.)
Edges removed	6 legacy kite-level exchanges (Batch Order Transfer, Sales Reporting, Marketing Lists, etc.)
Topology shift	ERP moves from hub (degree 5, highest centrality) to participant (degree 4, centrality shared with OMS)

This graph edit distance — 4 node changes, 16 edge changes — sizes the transformation at the landscape level. Each edge change implies integration work; each node change implies system procurement, build, or decommission.

Three structural shifts are immediately visible:

Decentralisation. The current state routes nearly all value through the ERP. The future state distributes integration across OMS, CDP, and direct eShop-to-WMS connections. The ERP's betweenness centrality drops significantly.

New direct paths. The current state has no direct connection between the web store and the warehouse — inventory checks route through the ERP. The future state introduces a direct eShop–WMS value exchange, reducing the shortest path for inventory visibility from 2 hops to 1.

Customer data separation. The current state embeds customer data management inside the CRM. The future state extracts it into a dedicated CDP with its own value exchanges, creating a clearer ownership boundary.

6. Transition Architecture via Phased Rewiring

The transformation from current to future state can be expressed as a sequence of value exchange cutovers — edges added and retired in phases. Each phase is a subgraph of the total graph edit.

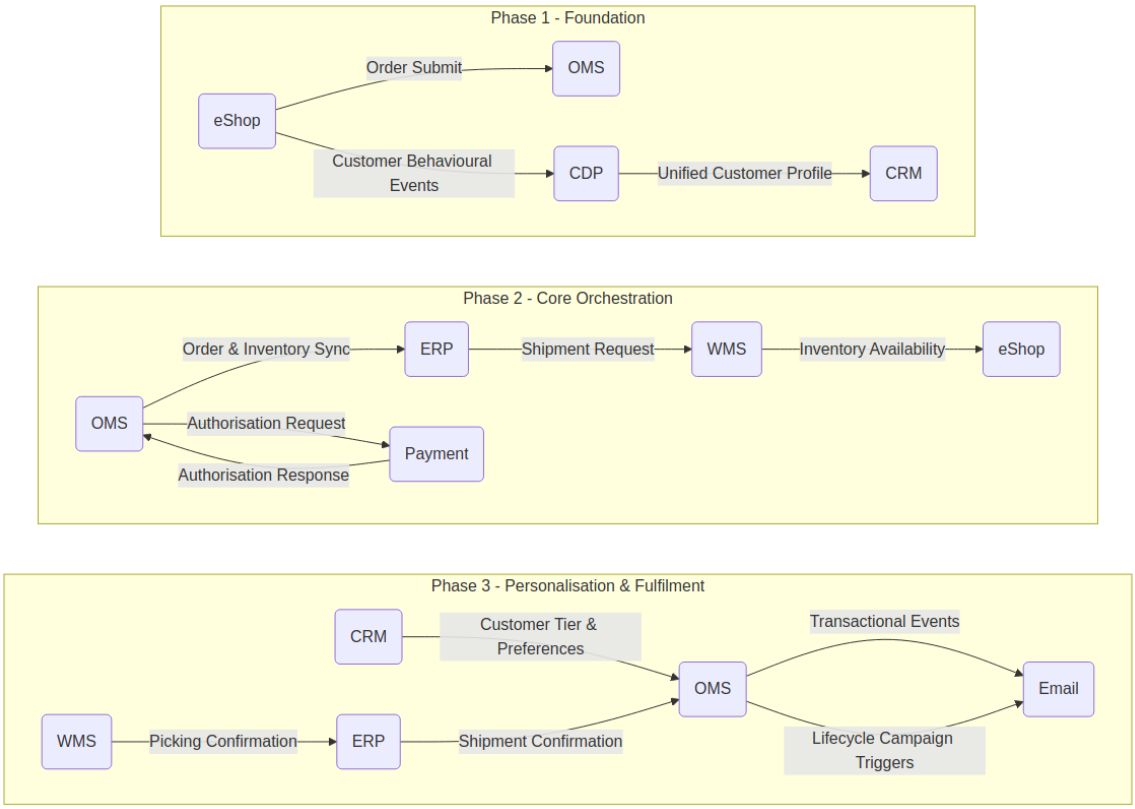

Fig. 4 — Transition roadmap as phased introduction of value exchanges.

Each phase can be analysed for transient broken paths. Before Phase 2 completes, inventory visibility from WMS to eShop does not exist — the legacy batch path through the ERP must remain operational as a temporary bridge. Identifying these transient dependencies is where the graph model earns its keep in programme planning: every edge that has been retired but whose replacement has not yet been activated is a risk.

7. Graph-Theoretic Analyses

By encoding the landscape as a directed graph with labelled, attributed edges, the following analyses become straightforward using standard graph libraries (NetworkX, igraph, Neo4j).

7.1 Structural Measures

Analysis	Graph Operation	Business Interpretation
Degree Centrality	Count of incident value exchanges per system.	Identifies integration-heavy systems that are expensive to change.
Betweenness Centrality	Frequency of a system lying on shortest value paths.	Reveals single points of failure in value flow — systems whose removal would disconnect parts of the landscape.
Shortest Path	Minimum hops between an event and an outcome.	Quantifies information latency and process complexity. Fewer hops generally means faster, more reliable delivery.
Graph Edit Distance	Node and edge changes between current and future state.	Sizes the transformation effort. Each edit implies integration, procurement, or decommission work.

7.2 Modularity and Clustering

Analysis	Graph Operation	Business Interpretation
Community Detection	Identify densely connected subgraphs (e.g., Louvain, label propagation).	Reveals natural system clusters — groups that exchange value intensively and should be managed together.
Modularity Score	Ratio of intra-cluster to inter-cluster edges.	Measures how cleanly the landscape separates into independent domains. Low modularity signals tightly coupled landscapes.
Bridge Identification	Find edges whose removal would disconnect communities.	Identifies critical integration points between domains — the value exchanges that, if disrupted, would isolate parts of the business.

7.3 Change and Resilience

Analysis	Graph Operation	Business Interpretation
Impact Propagation	Traverse all out-edges from a changed system to find affected downstream consumers.	Supports change-impact assessment and testing scopes.

Spectral Radius	Largest eigenvalue of the adjacency matrix.	A stability indicator: landscapes with high spectral radius are more susceptible to cascading failures. Reducing fan-out and improving modularity lowers this measure.
Altitude Coverage	Group sea-level exchanges by their kite-level parent; check that each kite-level exchange is fully realised.	Validates that strategic integration purposes are met by concrete value exchanges.

7.4 Weighted Extensions

The model can be extended with weighted edges for prioritisation and capacity analysis: business criticality (e.g., revenue-critical vs. operational), transaction volume (monthly message/event counts), and latency sensitivity (real-time vs. batch-tolerant). Weighted analyses shift the conversation from structural topology to operational risk.

8. Summary

Nodes are systems that refine business capabilities. Each system is a boundary where internal architecture meets the landscape. Its internal structure can be decomposed further but is treated as a unit at this level.

Edges are value exchanges — they describe what business value flows between systems, never how it is technically transported.

The model supports **two altitudes**: a kite-level view for strategic conversations and a sea-level view for architectural detail, with a symmetric refinement pattern connecting them.

Transitions are expressed as phased rewiring of value exchanges, enabling graph-theoretic cost estimation and transient risk analysis.

Graph analysis provides structural insight — centrality, modularity, spectral radius, impact propagation — that traditional EA diagrams cannot offer.

The entire model remains in the business domain, making it an effective communication tool across architects, business stakeholders, and engineers.

9. Getting Started

To apply this approach to your own landscape:

1. Start with kite-level edges. List your systems and the business purposes of each integration between them — one edge per purpose, no technical labels. This alone is often revealing.

2. Identify your highest-degree nodes. These are your integration-heavy systems. For each incident edge, ask: is this integration justified by a clear business outcome? Unjustified edges are candidates for retirement.

3. Run betweenness centrality. Find the systems that sit on the most value paths. These are your single points of failure and your highest-priority resilience concerns.

4. Detect communities. Look for clusters of systems that exchange value densely among themselves. These are natural programme boundaries — migrate, fund, or govern them together.

5. Decompose to sea level where it matters. You don’t need sea-level detail everywhere. Start with the highest-centrality systems and the value exchanges on your critical business paths.

Encode the graph in a structured format (JSON adjacency list, CSV edge list, or a graph database) and process it with standard libraries. The analyses described in Section 7 are all available out of the box in NetworkX, igraph, or Neo4j.