Decision Traces in the Modern Enterprise

An Ontology-Driven Architecture for AI-Ready Decision Intelligence in Logistics & Postal Services

> *A whitepaper exploring Business Ontology, Knowledge Graphs, Causal Graphs and Context Graphs as the foundation for explainable, traceable, AI-ready decision-making — framed in TOGAF, with BIZBOK, DDD, Data Mesh, APQC, SAP and ArchiMate as supporting lenses.*

Field	Value
Version	1.0
Date	May 2026
Audience	Business Analysts, Product Owners, Enterprise Architects (mixed technical levels)
Worked example	"PostNova plc" — a fictional national postal & parcel operator
Companion artefacts	`postnova-ontology.ttl` (Protégé-loadable), `README.md` (environment setup)
Reading time	~75 minutes (full); ~25 minutes (Part 1 only)

Reading Paths

This document is layered so that different audiences can extract value at different depths.

If you are a...	Read sections
Business Analyst / Product Owner	§0, §1, §3, §5, §6 (skim §4)
Enterprise / Solution Architect	§0–§9 in full; load the TTL
Data / AI Practitioner	§2.6, §3, §4.4, §6, §A–§C (queries)
Executive / Sponsor	§0, §3.1–§3.3, §9

Table of Contents

---

§0. Executive Summary

The strategic problem facing modern enterprises is no longer a shortage of data; it is a shortage of *defensible, traceable decisions* — decisions whose reasoning can be reconstructed, audited, replayed, and improved. Generative AI has compounded this problem: large language models routinely produce confident outputs whose underlying logic is opaque, whose context is shallow, and whose traceability to the enterprise's actual operating model is non-existent.

This paper argues that the cure is a **layered graph stack**:

1. A **Business Ontology** — the formal vocabulary of the enterprise, anchored to canonical artefacts (Business Model Canvas, Value Proposition Canvas, Operating Model Canvas).
2. A **Knowledge Graph** — instances and relationships expressed against that ontology, materialising the *what is* of the enterprise.
3. **Causal Graphs** — directed acyclic representations of *why things happen*, attached to business processes.
4. **Context Graphs** — bounded situational frames of *what is true right now*, wrapped around value streams and operating model elements.
5. **Digital Twins** — live, queryable mirrors of business capabilities, projecting all of the above into observable, simulatable state.

Stitched together with TOGAF's Architecture Development Method (ADM) as the sequencing backbone — and reinforced by BIZBOK's capability discipline, Domain-Driven Design's bounded contexts, Data Mesh's federated ownership, APQC's Process Classification Framework, SAP's industry models, and ArchiMate's notational rigour — this stack delivers what no single layer can: **decisions with traces**.

The worked case study is **PostNova plc**, a fictional national postal & parcel operator. We follow a single concrete decision — *"reroute Parcel PN-PCL-77821 via Birmingham instead of Manchester due to Storm Brendan"* — from its triggering event, through the rules, inputs, causal model, and situational context that produced it, all the way back to the value proposition and strategic objective the decision served. That trace, expressed in RDF/OWL and validated by SHACL, becomes the unit of explainability that auditors, regulators, customers and AI agents can all consume.

The deliverable accompanying this paper — `postnova-ontology.ttl` — is a working, Protégé-loadable ontology containing all the classes, properties, instance data, causal model and SHACL shapes used in the worked example.

> ***Headline claim.*** *The trillion-dollar opportunity in enterprise AI is not bigger models. It is richer context expressed as machine-readable graphs that mirror the real shape of the enterprise. The enterprise that*

> *can answer "why did we do that?" in milliseconds, with an evidentiary chain back to its operating model, will compound its decisions faster than competitors who treat AI as a black box.*

---

§1. Introduction: The Decision Visibility Crisis

§1.1 What is broken

Every large enterprise produces millions of decisions per day. A national postal operator reroutes parcels, prices contracts, hires drivers, opens lockers, retires sorting belts, accepts or declines insurance claims, and approves or denies refunds — all without a coherent record of *why*. Decisions are scattered across:

- Application logs (operational decisions, low-context).
- Spreadsheets and email threads (tactical decisions, no schema).
- Board minutes and slide decks (strategic decisions, no traceability).
- The heads of experienced operators (tacit knowledge, perishable).

The cost of this scattering is felt in three places:

1. **Audit & compliance** — regulators increasingly require evidence-grade explanations (EU AI Act, SR 11-7, GDPR Article 22).
2. **AI model governance** — every AI decision an enterprise makes inherits the explainability debt of its inputs.
3. **Organisational learning** — decisions that cannot be reconstructed cannot be evaluated; what cannot be evaluated cannot be improved.

§1.2 Why now: the context imperative

Two pieces of recent industry thinking sharpen the urgency. The first, from Foundation Capital, argues that the next great compounding asset for enterprise AI is the *context graph* — a richer object than the knowledge graph, encoding not just facts but the situational scaffolding that makes a fact *useful right now* (Foundation Capital, 2025 — see References §D). The second, from Modern Data 101, draws a sharp line between **analytics-ready data** (curated for human dashboards) and **AI-ready data** (curated for autonomous reasoning agents). The former is shaped by past questions; the latter must support questions that have not yet been asked (Modern Data 101, 2025).

Together they imply something uncomfortable for the enterprise data estate: *the warehouse and the lakehouse are not enough*. Tables, however well-modelled, encode **what** but not **why** and not **right-now-relevant**. To make AI decisions explainable and improvable at enterprise scale, the data plane needs to carry meaning, causation, and context — which means it needs to carry graphs.

§1.3 The thesis of this paper

> ***A decision is only as defensible as the trace behind it. A trace is only as rich as the graphs it walks. The graphs are only as useful as the ontology that gives them shared meaning.***

This paper translates that thesis into a working architecture, anchored in a logistics case study, and validated by a loadable ontology file.

§1.4 Audience and posture

The paper is written for a mixed audience of Business Analysts, Product Owners and Architects. The conceptual chapters (§2, §3) are paced for newcomers; the architecture chapter (§4) and the case study (§5–§6) descend into expert-level detail with SPARQL, SHACL, and DMN-style decision rules. Where a section is technical, a marginal **plain-language gloss** appears at the start.

---

§2. Conceptual Foundations

> ***Plain-language gloss.*** *This section gives precise definitions for ontology, knowledge graph, causal graph, context graph and decision model. If you are comfortable with these, skim to §3.*

§2.1 Ontology, Taxonomy, Folksonomy

An **ontology** is a *formal specification of a shared conceptualisation* (Gruber, 1993; Studer et al., 1998). Three claims hide inside that one sentence:

- **Formal** — written in a language with precise, computable semantics (RDF/OWL, first-order logic, UML class diagrams).
- **Specification** — explicit, versioned, addressable.
- **Shared** — partially agreed by stakeholders; never fully so. The discipline of ongoing alignment is *ontological commitment*.

This must be distinguished from two weaker artefacts often confused with it:

Artefact	Structure	Direction	Formal axioms?	Example
Folksonomy	Flat tags	Bottom-up	No	Twitter hashtags
Taxonomy	Tree (parent–child)	Top-down	No (only `is-a`)	Linnaean biology
Ontology	Graph (typed properties + axioms)	Bidirectional	Yes (logical)	OWL, SKOS-aligned schemas

An ontology subsumes a taxonomy (every well-formed taxonomy can be expressed as an ontology with only `rdfs:subClassOf` edges), but the reverse is not true. The bonus that ontology gives you is *named relationships beyond hierarchy*: `realises`, `supports`, `governedBy`, `appliesIn`. These named relationships are what lets a graph carry the shape of a business.

§2.1.1 The ontology stack

Following Guarino (1998) and the LEAD methodology, useful enterprise ontologies are **layered**:

Figure 2.1 — The Layered Enterprise Ontology Stack

The pragmatic value of layering is *reuse*: the foundational and core layers are written once and imported by every domain-specific ontology that follows. PostNova's domain ontology in our companion TTL imports the LEAD core; the application instance data imports the domain.

§2.1.2 Intensional vs extensional definitions

When you specify a class, you have two options:

- **Extensional** — list every member: `{Parcel, Letter, Pallet}`.
- **Intensional** — give axioms that decide membership: *"a Shipment is anything moved through the network with a tracking ID and a service level"*.

Enterprise ontologies should overwhelmingly favour **intensional** definitions. They are reasoner-friendly (an OWL DL classifier can infer membership), they do not break when a new sub-type appears, and they capture the genuine semantics of a class. The exception is closed reference data — country codes, ISO currencies, service-level enumerations — where extensional listing is appropriate.

§2.2 Knowledge Graphs

A **knowledge graph (KG)** is a graph-structured database where:

- Nodes are entities (instances of ontology classes).
- Edges are typed relationships (instances of ontology object properties).
- Edges and nodes can carry attributes (datatype properties).
- Some or all of the above is governed by an ontology, giving the graph a *schema with semantics* (not just a schema).

Two implementation traditions exist:

Tradition	Examples	Strengths
RDF / OWL	GraphDB, Stardog, Apache Jena, AnzoGraph	Open standards, formal semantics, OWL reasoning, SPARQL federation
Labelled property graphs (LPG)	Neo4j, TigerGraph, Memgraph	Performance on traversal, intuitive modelling, Cypher/GQL

The PostNova reference architecture uses **RDF/OWL** because traceability and reasoning are first-class requirements. RDF-star (RDF 1.2) closes the historical gap on edge properties, so the LPG advantage for analytics-style queries is narrower than it once was.

> ***Vector-only RAG is a knowledge graph manqué.*** *A retrieval-augmented generation pipeline that retrieves text chunks by cosine similarity is doing knowledge graph work* ***without the graph****: it is recovering relationships from co-occurrence rather than asserting them as first-class typed edges. This is fine for assistive search; it is insufficient for enterprise decision traceability.*

§2.3 Causal Graphs

A **causal graph** is a directed acyclic graph (DAG) in which each node represents a variable and each edge represents an asserted causal mechanism (Pearl, 2009). Three operations distinguish causal graphs from correlational ones:

1. **Observation** — `P(Y | X)`, the conditional probability of Y given that X is observed.
2. **Intervention (Pearl's `do`-operator)** — `P(Y | do(X))`, the probability of Y if we *force* X to a value, breaking incoming arrows to X.
3. **Counterfactual** — *"what would Y have been, had X been different, holding everything else constant?"*

Why does an enterprise care? Because most operating decisions are **interventions**, not observations. *"If we add a hub at Birmingham, what happens to delivery times?"* is a `do`-question, not a correlation question. Causal graphs let the enterprise simulate interventions without running the experiment.

In §5.4 we attach a causal graph to PostNova's *Dynamic Routing Process*. The model has 7 nodes — Storm Severity, Hub Capacity, Route Congestion, Transit Time, SLA Breach Probability, Customer Satisfaction, Reroute Decision — and edges with typed mechanisms (`mechanismDescription`) and signed strengths (`edgeStrength`).

§2.4 Context Graphs

A **context graph** is a graph that captures the *situational scaffolding* surrounding a decision or process — the elements that change the meaning of facts in the knowledge graph without being facts about the enterprise itself.

The distinction matters. Consider the assertion:

> *"PostNova's Manchester hub processes 6,000 parcels/hour."*

That is a knowledge-graph fact: a stable property of an entity. Now consider:

> *"PostNova's Manchester hub is currently running at 35% throughput because Storm Brendan is overhead and three sortation belts are offline; the disruption window closes at 03:00; a similar storm in 2024 took*

> *14 hours to clear."*

That is a **context graph** assertion. It binds: a temporal window, an external weather event, an operational state, a regulatory/policy frame (driver hours), and a piece of tacit institutional knowledge (the 2024 analogue). Take any one of those elements away and the routing decision changes.

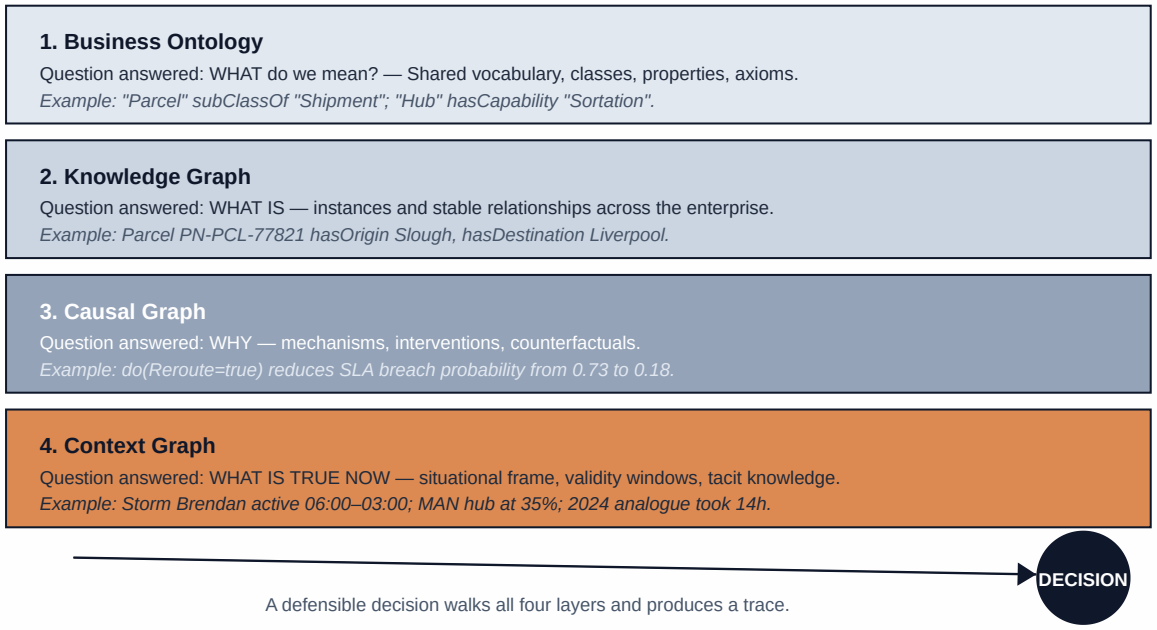
In RDF terms, the difference is often expressed using **named graphs** or **RDF-star quoted triples**: a knowledge-graph triple is asserted unconditionally; a context-graph triple is asserted *within* a context, with validity windows, sources, and confidence. The ontology in our companion TTL models context with explicit `ctx:Context` and `ctx:ContextElement` classes plus `ctx:validFrom` / `ctx:validTo` datatype properties.

The Foundation Capital piece on context graphs makes a stronger claim worth engaging with: that context — not parameter count and not data volume — is where enterprise AI value will compound (Foundation Capital, 2025). Their framing is consistent with what enterprise architects have been observing for a decade: **the bottleneck on intelligent automation is meaning, and meaning lives in context.**

§2.5 Putting the four together

Figure 2.5 — The Four-Graph Decision Trace Stack

Each graph adds a question that the layer below cannot answer.



A defensible decision walks all four layers and produces a trace.

A decision becomes *traceable* when it can be replayed across all four layers: which classes it concerned (ontology), which instances it touched (KG), which mechanism it invoked (causal), and which situation framed it (context). All four are necessary; any one alone is degenerate.

§2.6 Decision Models and DMN

The Decision Model and Notation standard (DMN 1.4, OMG) provides a complementary modelling discipline for the decision layer itself. A DMN model decomposes a decision into:

- **Decision** — the act of selecting an outcome.
- **Input Data** — the values consumed.
- **Knowledge Source** — the authority (regulation, policy, expert) governing the decision.

- **Business Knowledge Model** — the reusable logic (decision table, FEEL expression, ML model) applied.

Our ontology mirrors DMN's vocabulary explicitly: `dec:Decision`, `dec:DecisionInput`, `dec:DecisionRule`, `dec:DecisionOutcome`. The benefit of expressing DMN constructs *within* the same ontology that holds the business model is that the decision is no longer a free-floating BPMN footnote — it is wired to the strategic objective it serves, the capability that owns it, and the value proposition it protects. That wiring is what makes a trace useful.

§2.7 Provenance: the W3C PROV-O model

The final foundational piece is **provenance**. The W3C PROV-O ontology gives us three core classes (`Entity`, `Activity`, `Agent`) and properties such as `wasGeneratedBy`, `wasAssociatedWith`, `wasDerivedFrom`. We align our `dec:Decision`, `dec:DecisionTrace` and `dec:DecisionMaker` classes with PROV's `Activity`, `Entity` and `Agent` respectively. This single alignment is what makes our decision traces interoperable with any modern data lineage or audit toolchain.

---

§3. Part 1 — The Case for an Enterprise Ontology + Context Graph Stack

> ***Plain-language gloss.*** *Why bother with all four graphs? Because the alternatives — vector RAG alone, dashboards alone, BPMN alone — each fail in a way that the stack collectively repairs. This section is the business case.*

§3.1 The decision black box

Before defending the stack, name the disease. Decisions in most enterprises today are made in one of four ways:

1. **Hard-coded in applications** — opaque to anyone outside engineering, scattered across systems.
2. **Modelled in BPMN diagrams** — visible to architects, but disconnected from the data that the decision actually consumes.
3. **Embedded in trained ML models** — predictive but not explanatory; SHAP values describe local feature importance, not enterprise meaning.
4. **Held in operator heads** — the most context-rich source, the most fragile, and the source most threatened by retirement, attrition and AI substitution.

When a decision goes wrong (a parcel misses SLA, a refund is wrongly denied, a contract is mispriced) the post-mortem requires reconstructing all four sources and stitching them together by hand. This is expensive when done by humans and impossible when done at AI velocity. **The graph stack collapses that reconstruction effort to a SPARQL query.**

§3.2 AI-ready data versus analytics-ready data

Modern Data 101's distinction between AI-ready and analytics-ready data is worth restating in the language of decision traces (Modern Data 101, 2025). Analytics-ready data has been *shaped by past questions*: the dimensions, hierarchies, and aggregations match the dashboards leadership has historically asked for. AI-

ready data must support *questions not yet asked*, including questions in the form of `do`-interventions or counterfactuals.

Concretely, AI-ready data needs five properties that traditional dimensional models do not require:

Property	Why AI needs it	Where the four-graph stack supplies it
Semantics	Agents must reason about *what a column means*, not infer it from a dashboard label.	Business Ontology
Relationships	Agents must traverse, not just join, to reach related entities.	Knowledge Graph
Causality	Agents must distinguish correlation from causation when proposing actions.	Causal Graph
Context	Agents must condition on the situation, not just the row.	Context Graph
Provenance	Agents must justify outputs with auditable derivation chains.	PROV-O on the Decision Trace

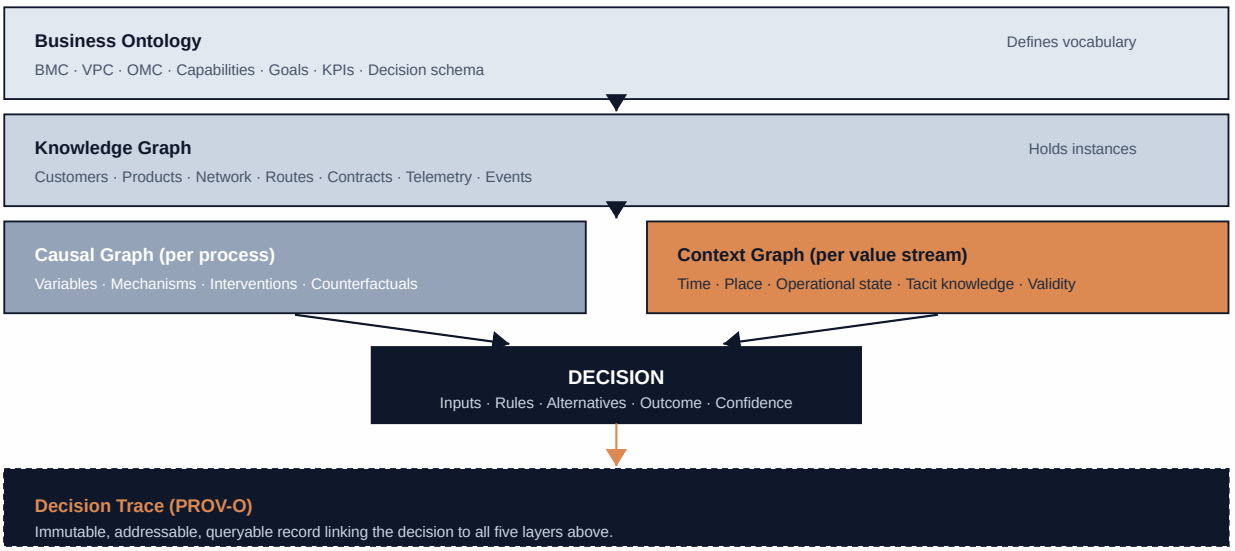
A vector index over an unstructured document corpus supplies *none* of these in a structured form. It supplies a fifth thing — *unstructured semantic recall* — which is genuinely useful but cannot bear the weight of audited, governed enterprise decisions.

§3.3 The decision trace framework

We can now state the framework that the rest of the paper develops:

Figure 3.3 — The Decision Trace Framework

Five layers, one trace; every decision is reconstructible.



The shape to internalise: **two horizontal layers (ontology + KG) define the enterprise's stable knowledge; two vertical layers (causal + context) wrap individual processes and value streams; one trace at the bottom binds them to a specific decision instance.** A decision is well-formed if and only if it has at least one edge to each layer. SHACL shapes (Appendix B) enforce this invariant.

§3.4 Why this beats vector-only RAG

Vector-only retrieval-augmented generation is the dominant pattern in 2025–2026 enterprise AI deployments. It is also, almost universally, the wrong primary substrate for *decisions* (as opposed to assistive search). Five specific failures:

1. **No type system.** A vector index does not know that `Parcel` and `Shipment` are related by subclassing; it only knows their text co-occurs. Semantic drift is invisible.
2. **No traversal.** *"Find all parcels affected by Storm Brendan whose senders are gold-tier customers"* is a graph traversal. RAG approximates this with re-ranking, badly.
3. **No causality.** A vector index will happily retrieve correlated artefacts. Pearl's hierarchy of causation (observation < intervention < counterfactual) is invisible to it.
4. **No validity windows.** The chunk that says *"MAN hub processes 6,000 parcels/hour"* does not know that today the chunk is wrong.
5. **No provenance.** Outputs cannot be attributed to specific assertions made at specific times by specific agents.

The pattern that fixes this is sometimes called **GraphRAG**: a hybrid where a knowledge/context graph supplies typed retrieval and structured reasoning, and the vector index supplies unstructured semantic recall over documents. The graph is the spine; the vector index is connective tissue.

§3.5 The strategic argument

Stepping up a level: the enterprise that builds this stack is buying three compounding assets at once.

- **An institutional memory** that survives staff turnover.
- **A reasoning surface** that AI agents can use as a source of truth rather than fabricating from training data.
- **An audit trail** that turns regulatory exposure into a feature rather than a liability.

None of these accrue to a vector-only deployment. All three accrue to a graph-stack deployment from day one of having any meaningful content in the graph. This is the trillion-dollar argument restated in enterprise-architecture terms (Foundation Capital, 2025).

---

§4. Part 2 — Architecture using TOGAF (with Supporting Lenses)

> ***Plain-language gloss.*** *This section maps the four-graph stack onto TOGAF's Architecture Development Method, then layers in BIZBOK, DDD, Data Mesh, APQC, SAP and ArchiMate where each adds value. If you are a working architect, this is your blueprint.*

§4.1 Why TOGAF as the primary frame

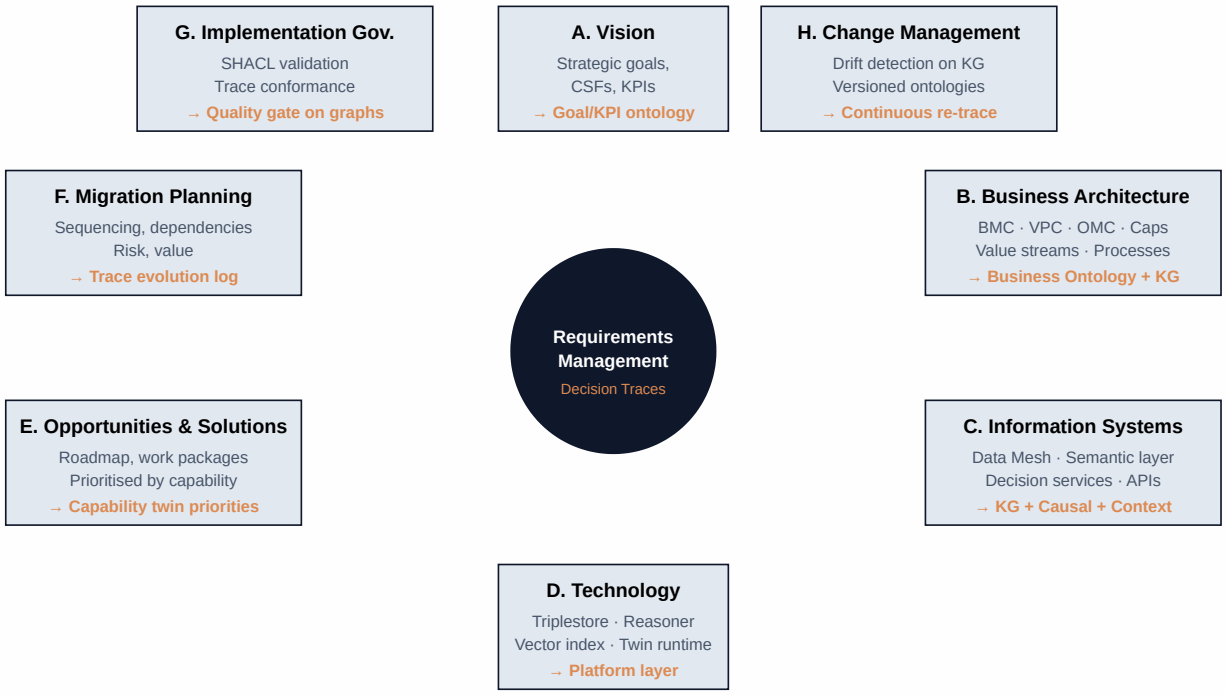
TOGAF's Architecture Development Method (ADM) is selected as the primary frame for one pragmatic reason: its phase sequence (Vision → Business → Information Systems → Technology → Migration → Implementation Governance) maps cleanly onto the construction order of the graph stack. Phase B (Business Architecture) is where the Business Ontology must be specified; Phase C (Data + Application) is where the

Knowledge, Causal, and Context graphs are materialised; Phase D (Technology) is where the graph platform, reasoner and digital twin runtime are selected.

TOGAF is sometimes critiqued for being process-heavy. The reply is that *the ADM is a checklist of questions, not a waterfall*; modern teams iterate it in increments aligned with capability releases, not annual cycles.

Figure 4.1 — TOGAF ADM with the Graph Stack Overlay

Each ADM phase produces or consumes specific graph artefacts.



§4.2 Phase A — Architecture Vision

Phase A produces the strategic narrative: goals, CSFs, KPIs, stakeholder map. In the graph stack, Phase A's output is a small set of `lead:Goal` and `lead:KPI` instances that anchor every downstream decision trace. Without this anchor, a decision can be valid-but-purposeless: well-traced inputs and rules, but no claim about *why* the enterprise made it.

Concrete deliverable in PostNova: the instance `pn:Obj-OnTimeDelivery98` ("Achieve 98% On-Time Delivery") plus KPIs `pn:KPI-OTDPercent` and `pn:KPI-CostPerParcel`. Every decision in §6 will trace to one of these.

§4.3 Phase B — Business Architecture

Phase B is where the Business Ontology is specified and the Knowledge Graph schema is committed. This phase loads heavily on three canonical artefacts:

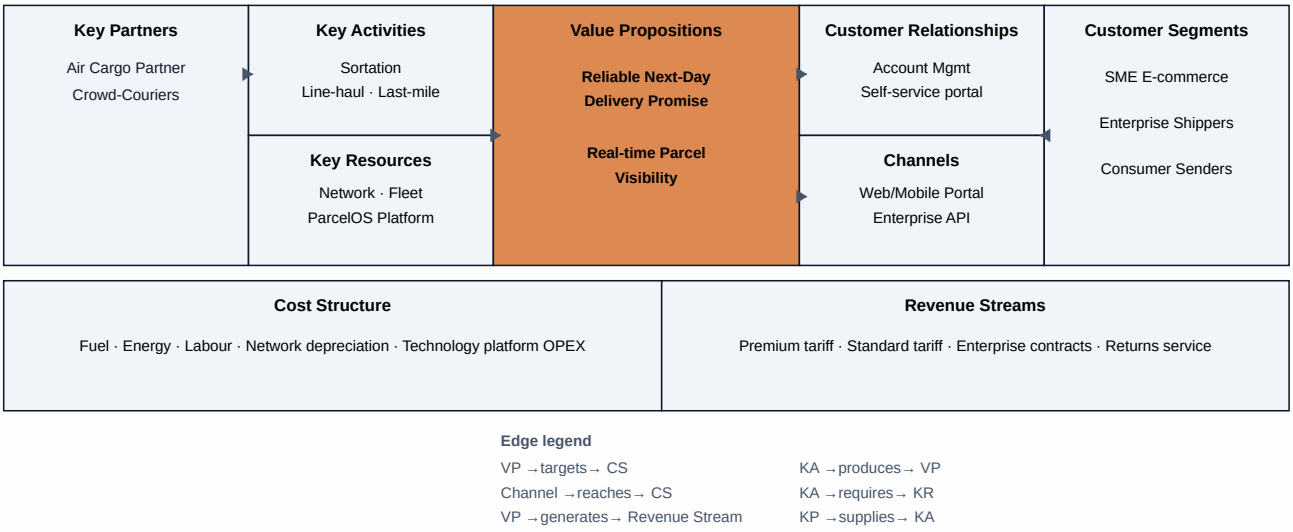
§4.3.1 The Business Model Canvas as ontology

Osterwalder and Pigneur's Business Model Canvas (BMC) is conventionally a 9-block sketch on a wall. Treated as an ontology, it becomes nine top-level classes (`bmc:CustomerSegment` , `bmc:ValueProposition` , `bmc:Channel` , `bmc:CustomerRelationship` , `bmc:RevenueStream` , `bmc:KeyResource` , `bmc:KeyActivity` ,

`bmc:KeyPartner`, `bmc:CostStructure`) and a fixed set of typed relationships between them (`targets`, `reaches`, `produces`, `requires`, `generates`).

Figure 4.3 — BMC as a Knowledge Graph (PostNova instance overlay)

Nine canonical blocks become nine classes; their relationships become typed edges.



The benefit of treating the BMC as ontology is twofold. First, the canvas becomes *queryable*: *"show me all key activities that produce the next-day value proposition"* is a SPARQL pattern. Second, it becomes *traceable*: a decision can declare its supporting `bmc:ValueProposition`, and the trace can show the canvas block at risk if the decision fails.

§4.3.2 The Value Proposition Canvas as ontology

The Value Proposition Canvas (VPC) extends the BMC by decomposing each value proposition into a `vpc:CustomerProfile` (Jobs, Pains, Gains) and a `vpc:ValueMap` (Products/Services, Pain Relievers, Gain Creators). The fit between the two is expressed by the object property `vpc:fits`. In our ontology this becomes:

```
pn:VM-NextDay vpc:fits pn:CP-SMEEcommerce ;
              vpc:relieves pn:Pain-LateDelivery, pn:Pain-LackVisibility ;
              vpc:creates pn:Gain-CustomerLoyalty .
```

This is what gives the knowledge graph its *teeth*: every decision that protects the next-day value proposition can be traced to the specific customer pain it relieves and the customer gain it creates.

§4.3.3 The Operating Model Canvas as context graph anchor

Andrew Campbell's Operating Model Canvas (OMC) decomposes operations into six elements (POLISM): **P**rocesses, **O**rganisation, **L**ocations, **I**nformation, **S**uppliers, **M**anagement systems. We model each as a class (`omc:OMCProcessElement`, `omc:OMCOrganisation`, etc.). The OMC is the natural anchor for **context graphs** because every context element binds to one or more OMC blocks: a weather event affects Locations; a regulatory change affects Management Systems; a strike affects Organisation.

§4.3.4 BIZBOK capabilities

BIZBOK insists on a clean separation between *capabilities* (what the enterprise does) and *processes* (how it does it). We honour this with `cap:Capability` as a distinct class hierarchy, decomposed L1 → L2 → L3, and connected to processes by `cap:realizes`. PostNova's L1 capabilities include Network Planning, Sortation, Line-Haul, Last-Mile, Customer Communications, Exception Management. Each L1 has a corresponding **Capability Digital Twin** (§5.6).

§4.3.5 APQC Process Classification Framework

For process taxonomy, the APQC Process Classification Framework (PCF) provides a defensible cross-industry baseline. We do not import the entire PCF; instead, we tag PostNova's `proc:Process` instances with their PCF banner reference (e.g., 4.3.4 — *"Schedule and dispatch fleet"*). This gives the enterprise a defensible benchmarking surface.

§4.3.6 SAP industry models

SAP's industry reference models for Postal & Logistics (S/4HANA Industry Cloud for Logistics) supply concrete entity definitions — Shipment, ShipmentItem, TransportRoute, HandlingUnit — which we align by `skos:closeMatch` rather than direct subclassing. This lets PostNova interoperate with SAP-standard data products without surrendering the ontology's structural independence.

§4.3.7 ArchiMate as visual grammar

ArchiMate 3.2's Business Layer notation is the visual register for stakeholder communication. The mapping is straightforward: `cap:Capability` → ArchiMate Capability element; `proc:Process` → Business Process; `lead:Actor` → Business Actor; `lead:Goal` → Goal (motivation extension). Architects can render slices of the ontology as ArchiMate diagrams without semantic loss.

§4.4 Phase C — Information Systems Architecture

Phase C splits into Data Architecture and Application Architecture. The graph stack lives primarily in Data, with consuming services in Application.

§4.4.1 Data Architecture: Data Mesh + the Semantic Layer

Data Mesh's principle of *domain-aligned data products with federated governance* aligns naturally with the ontology's layered structure (Dehghani, 2022). PostNova's data products are organised by capability domain — **Sortation Data Product**, **Line-Haul Data Product**, **Customer Data Product**, etc. Each product is owned by the corresponding capability team. The ontology supplies the **federated computational governance** layer: a shared vocabulary that every data product must speak, even though the product internals can vary.

Within each domain, the architecture has three tiers:

1. **Operational tier** — application databases, event streams (Kafka), telemetry. Source of truth for current state.
2. **Semantic tier** — RDF/OWL graph store (e.g., GraphDB, Stardog, AnzoGraph) ingesting from the operational tier through R2RML / RML mappings; this is where the Knowledge, Causal and Context graphs materialise.

3. **Consumption tier** — SPARQL endpoints, GraphQL adapters, vector indexes for unstructured content, BI marts. The decision services in Application Architecture call into this tier.

The non-obvious move is that the **semantic tier is where the Decision Trace store also lives**. Traces are RDF entities, governed by the same SHACL shapes as the rest of the graph. This collapses what would otherwise be a separate audit/lineage product into the data plane.

§4.4.2 Domain-Driven Design: bounded contexts as semantic boundaries

Eric Evans' Domain-Driven Design supplies the missing piece on *boundary management*. A single enterprise ontology covering everything is brittle and expensive to govern. Instead, each capability domain has a **bounded context** (DDD term) within which its model is canonical, with **context maps** declaring how it relates to neighbouring contexts. In RDF, bounded contexts become **named graphs**; context maps become **owl:equivalentClass / skos:closeMatch / owl:sameAs** assertions across them.

This is what makes the architecture survive at enterprise scale: nobody owns the whole graph; everybody owns their slice; the relationships are explicit.

§4.4.3 Application Architecture: decision services, context engine, twin runtime

Three logical services dominate Application Architecture:

- **Decision Service** — exposes decisions as APIs (REST/gRPC). Internally evaluates DMN tables / FEEL expressions / ML models, persists the trace, returns the outcome.
- **Context Engine** — assembles the relevant context graph for a given decision request. Subscribes to weather, capacity, regulatory and operational event streams; maintains validity windows; exposes context as a graph fragment.
- **Digital Twin Runtime** — projects capability twins, maintains state, feeds telemetry into causal models for what-if simulation.

These three communicate via RDF graph fragments, not via opaque JSON payloads. The economy of a graph-native API is significant: every payload self-describes against the ontology, every consumer can validate against SHACL, and every cross-service join is a graph traversal rather than a fragile correlation key.

§4.5 Phase D — Technology Architecture

The technology stack reduces to a small number of principled choices.

Layer	Choice	Rationale
Triplestore	RDF 1.2 / OWL 2 store with SPARQL 1.1 (e.g., GraphDB, Stardog, Apache Jena Fuseki, AnzoGraph)	Open standard, federation-friendly
Reasoner	Embedded OWL DL reasoner (HermiT, Pellet, ELK) for offline classification; rule layer (SHACL Advanced, SWRL, or Datalog) for online	DL gives schema validity; rules give business semantics
Validation	SHACL (W3C standard)	Constrains shapes of the graph; Appendix B
Vector index	Co-located vector store (e.g., Weaviate, Qdrant, pgvector) for unstructured retrieval	GraphRAG hybrid
Streaming	Kafka + Kafka Streams for event ingest into the graph	Maintains validity windows on context graph
Twin runtime	Time-series store (InfluxDB / TimescaleDB) + simulator harness	Capability twin telemetry + what-if execution
Query gateway	GraphQL adapter over SPARQL for application teams unfamiliar with SPARQL	Adoption friction reduction
Modelling tooling	Protégé for ontology authoring; SHACL Playground for shape testing; Cube/Y for visualisation	Standard practitioner toolchain

§4.6 Phase G — Implementation Governance

Implementation Governance becomes radically simpler when the architecture's deliverables are graphs. The conformance checks reduce to SHACL validation runs on each release. Appendix B gives the canonical SHACL shapes; the most important is `pn:DecisionTraceShape`, which mandates that every `dec:Decision` instance has at least one input, one rule, one outcome, one context, one supported objective, and one trace.

Failed validation is failed governance. There is no longer a separate "documentation gate" — the documentation *is* the graph.

---

§5 — Part 3: The PostNova Case Study

§5.1 Company profile

PostNova plc is a fictional national postal and parcel operator used as the running example. Operating profile:

- Annual volume: ~1.2bn parcels
- Network: 4 national hubs (London/Heathrow, Manchester, Birmingham, Glasgow), 28 regional sortation depots, ~1,400 last-mile rounds
- Workforce: ~38,000 (sortation, drivers, customer-care, network planning, technology)
- Strategic posture: defending universal-service obligation while growing a profitable B2B and SME e-commerce parcels business against well-capitalised carrier competition
- Decision pressure: weather, fuel volatility, industrial relations, regulator scrutiny, AI-routing competition

The canonical decision we will trace end-to-end is an operational one with strategic implications: **Storm Brendan is forecast to land over the Manchester hub during the morning of 9 May 2026; should parcel PN-PCL-77821 (and the ~140k parcels in its cohort) be rerouted via Birmingham?**

§5.2 The Business Ontology, on the Business Model Canvas

PostNova's BMC is encoded in the ontology (see the TTL `bmc:` namespace and `pn:` instances). The nine BMC blocks become the top-level scaffolding:

- **Customer Segments**: SME e-commerce sellers (CS-SMEEcommerce), enterprise contract logistics, consumer postal users
- **Value Propositions**: Next-Day Promise (VP-NextDayPromise), Real-time Visibility (VP-RealtimeVisibility), Universal Service (VP-UniversalService)
- **Channels**: Web/API, contact centre, retail counters, partner marketplaces
- **Customer Relationships**: SLA-governed (B2B), self-service (consumer)
- **Revenue Streams**: per-parcel postage, contract revenue, surcharges, returns
- **Key Resources**: the logistics network itself (KR-LogisticsNetwork), brand, regulated licence, workforce, data
- **Key Activities**: Sortation (KA-Sortation), Line-Haul (KA-Linehaul), Last-Mile (KA-LastMile), Network Planning, Customer Communications
- **Key Partners**: carrier subcontractors, fuel suppliers, postal regulators, retail partners
- **Cost Structure**: fleet & fuel, labour, facilities, technology, regulatory levies

The graph encoding is deliberate: each BMC block is not just a label on a slide, it is a class with instances, properties and relations. The Next-Day Promise (`pn:VP-NextDayPromise`) is `vpc:fits` linked to SME E-commerce customer profile, `bmc:hasRevenueStream` linked to per-parcel postage, and `bmc:supportedBy` linked to Sortation, Line-Haul and Last-Mile activities. The whole BMC is a navigable graph, not a static artefact.

§5.3 The Knowledge Graph, on the Business Model Canvas + Value Proposition Canvas

The Knowledge Graph layer adds three things to the BMC ontology:

1. **VPC fitment edges** — every `vpc:ValueMap` is linked to its `vpc:CustomerProfile` via `vpc:fits` , with the specific pains relieved and gains created enumerated as instances. PostNova's `pn:VM-NextDay` relieves `pn:Pain-LateDelivery` and `pn:Pain-LackVisibility` , and creates `pn:Gain-CustomerLoyalty` .
2. **Instance-level facts** — the actual customer accounts, contracts, parcel volumes, SLA targets, hub capacities. This is where the analytics-ready feed (claim counts, on-time percentages) would have stopped; the KG keeps going into the relationships.
3. **Cross-block joins** — facts that span BMC blocks: *"Customer Segment SME E-commerce contributes 38% of Revenue Stream per-parcel postage, depends on Key Activity Sortation, and is most affected by Cost Structure fleet & fuel."* These joins are first-class edges in the graph, not derived joins in a BI tool.

The non-obvious benefit: every business-strategy question — *"if we lose Sortation hub MAN for 6 hours, which Customer Segments take the biggest revenue hit?"* — becomes a SPARQL traversal over the BMC+VPC graph. There is no separate "strategy model" to maintain.

§5.4 Causal Graphs on Business Processes

Operational processes generate the causal graphs. The Order-to-Doorstep value stream has six stages (Place → Inject → Sort → Line-Haul → Final-Mile → Deliver). Each stage has measurable variables — capacity utilisation, throughput, dwell time, exception rate — and these variables exhibit causal structure that is *invariant under intervention* in the Pearl sense.

The causal model for the weather-disruption decision is shown below.

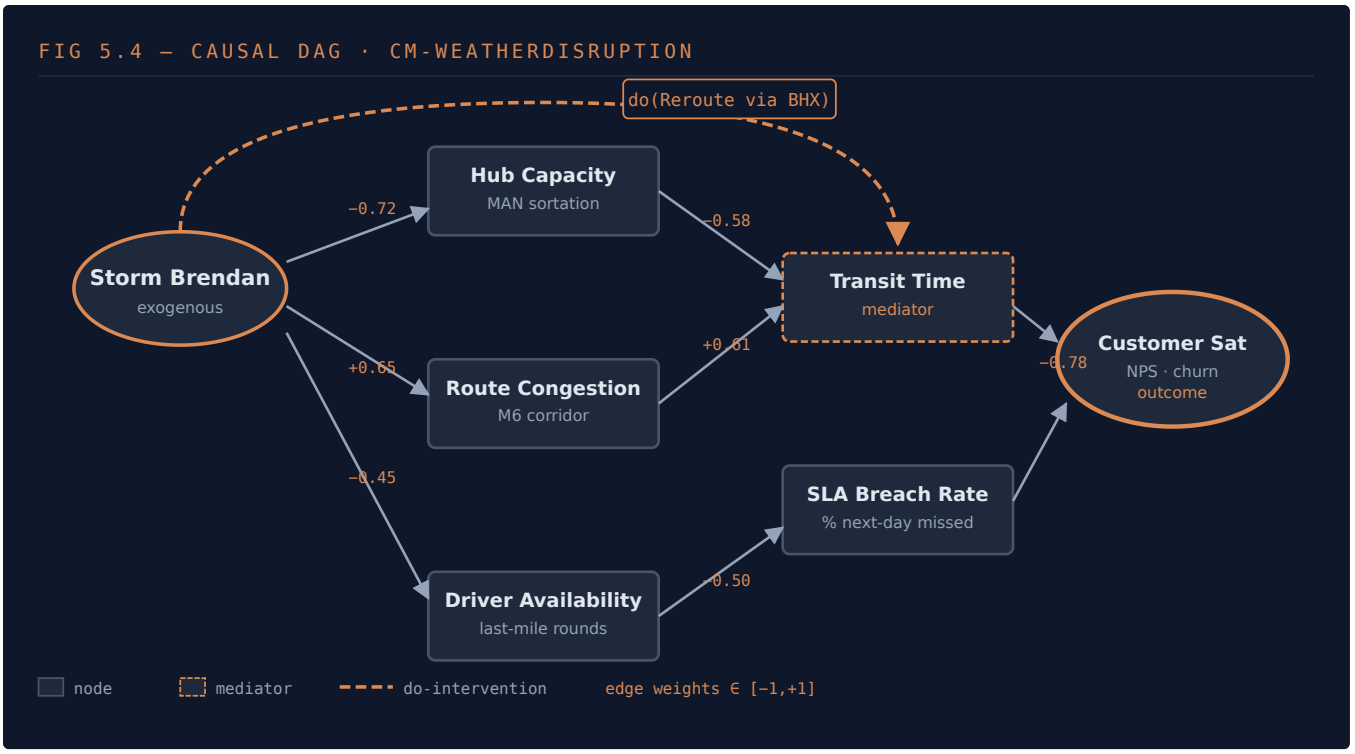


Figure 5.4 — *Causal DAG `pn:CM-WeatherDisruption`. Storm is the exogenous root cause; Transit Time is the mediator carrying most of the effect to the Customer Satisfaction outcome. The dashed edge is Pearl's `do()` intervention: forcing Reroute-via-Birmingham severs the Storm → Hub-Capacity link for the affected cohort, breaking the causal chain at its strongest edge (−0.72).*

The DAG is encoded in the TTL as `pn:CM-WeatherDisruption` with seven `cau:CausalNode` instances and six `cau:CausalEdge` instances, each carrying a `cau:strength` literal in [−1, +1]. The strengths are not point estimates from a single model — they are the current best estimate from a counterfactual estimator that the Twin Runtime maintains.

The causal graph is what enables the **counterfactual question**: *"What would Customer Satisfaction have been if we had not rerouted?"* That question is answered by a `cau:Counterfactual` instance that fixes the storm and the rest of the world to actuality but sets `cau:Reroute = false`, then recomputes the outcome along the DAG. The counterfactual is stored alongside the actual outcome in the decision trace.

§5.5 The Context Graph on the Operating Model Canvas + Value Streams

The Order-to-Doorstep value stream is the spine; the OMC blocks are the ribs; context elements wrap both.

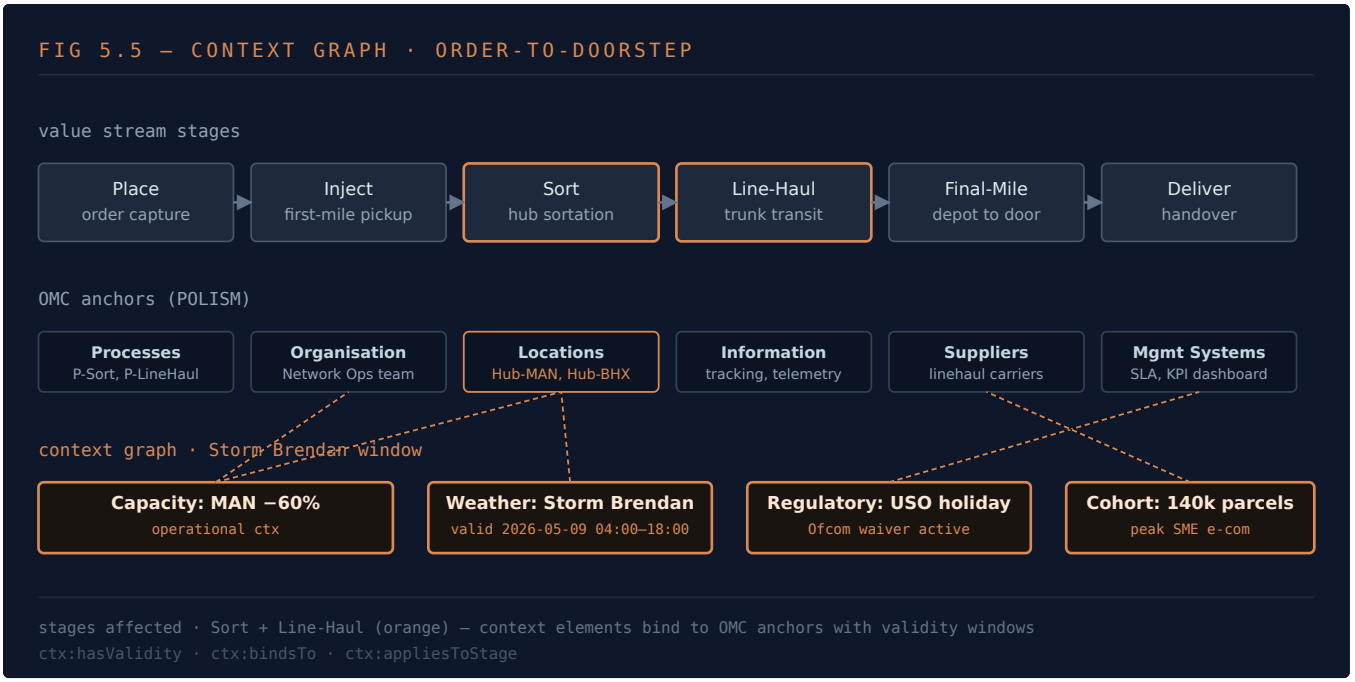


Figure 5.5 — *The context graph for the storm window. Each context element (orange) binds to one or more OMC anchors via `ctx:bindsTo`, carries a temporal validity window via `ctx:hasValidity`, and applies to specific value-stream stages (Sort and Line-Haul are shaded orange). When the validity window closes, the context element is automatically deactivated; the decision trace retains the binding for replay.*

What this gives the architecture is **scoped context**. The decision service does not retrieve "all the world's weather data"; it asks the context engine for the context graph fragment that is currently valid for the affected OMC blocks of the affected value-stream stages. That fragment is, in this case, four context elements. Small, traceable, replayable.

§5.6 Digital Twins on Business Capabilities

Digital twins in PostNova are organised around **capabilities**, not individual assets. The reasoning: an asset twin (a single sortation machine) is too narrow to carry strategic-decision context; an enterprise twin is too broad. A capability twin sits at the right granularity — it represents the *function* the enterprise performs, with all the assets, processes, people and information that compose it.

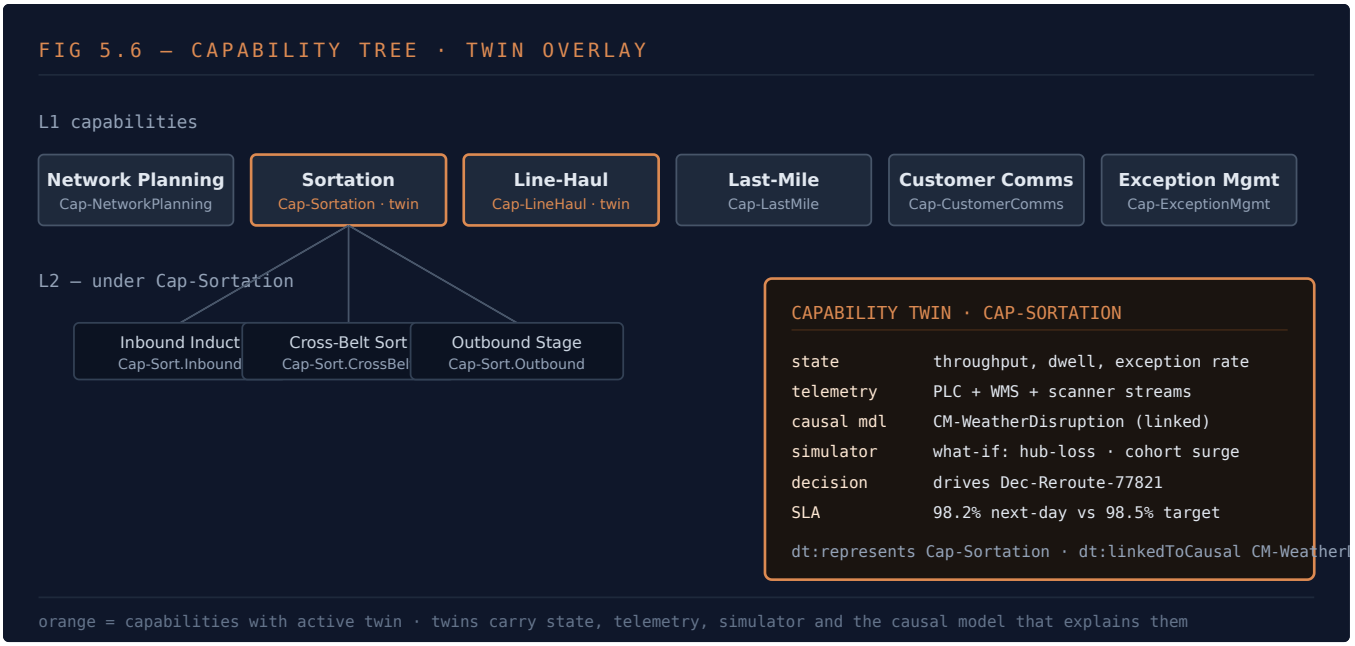


Figure 5.6 — *Capability tree (L1 → L2) with twin overlay. The Sortation and Line-Haul capabilities have active digital twins; the twin maintains state (throughput, dwell, exception rate), pulls telemetry from PLC and WMS streams, holds a reference to the causal model that explains its dynamics, runs a simulator for what-if (hub-loss, cohort surge), and is the addressable object that the decision service queries when deciding to reroute.*

The crucial design pattern is the **capability twin → causal model** edge. The twin is not a simulator alone; it is a simulator that *knows what causes what* via its linked `cau:CausalModel`. This is what allows the simulator to answer *"if we lose Hub-MAN for 6 hours, what is the SLA breach rate, and which customer segments are most exposed?"* — a question no purely statistical twin can answer because it requires intervention semantics.

---

§6 — Part 4: End-to-End Decision Trace Walkthrough

We now walk the entire stack for a single decision: *"Reroute Parcel PN-PCL-77821 (and its 140k-parcel cohort) via Birmingham instead of Manchester, on the morning of 9 May 2026, in response to Storm Brendan."*

The decision is `pn:Dec-Reroute-77821` in the TTL. We trace it across the four graphs in the order the runtime walks them.

§6.1 The trace path

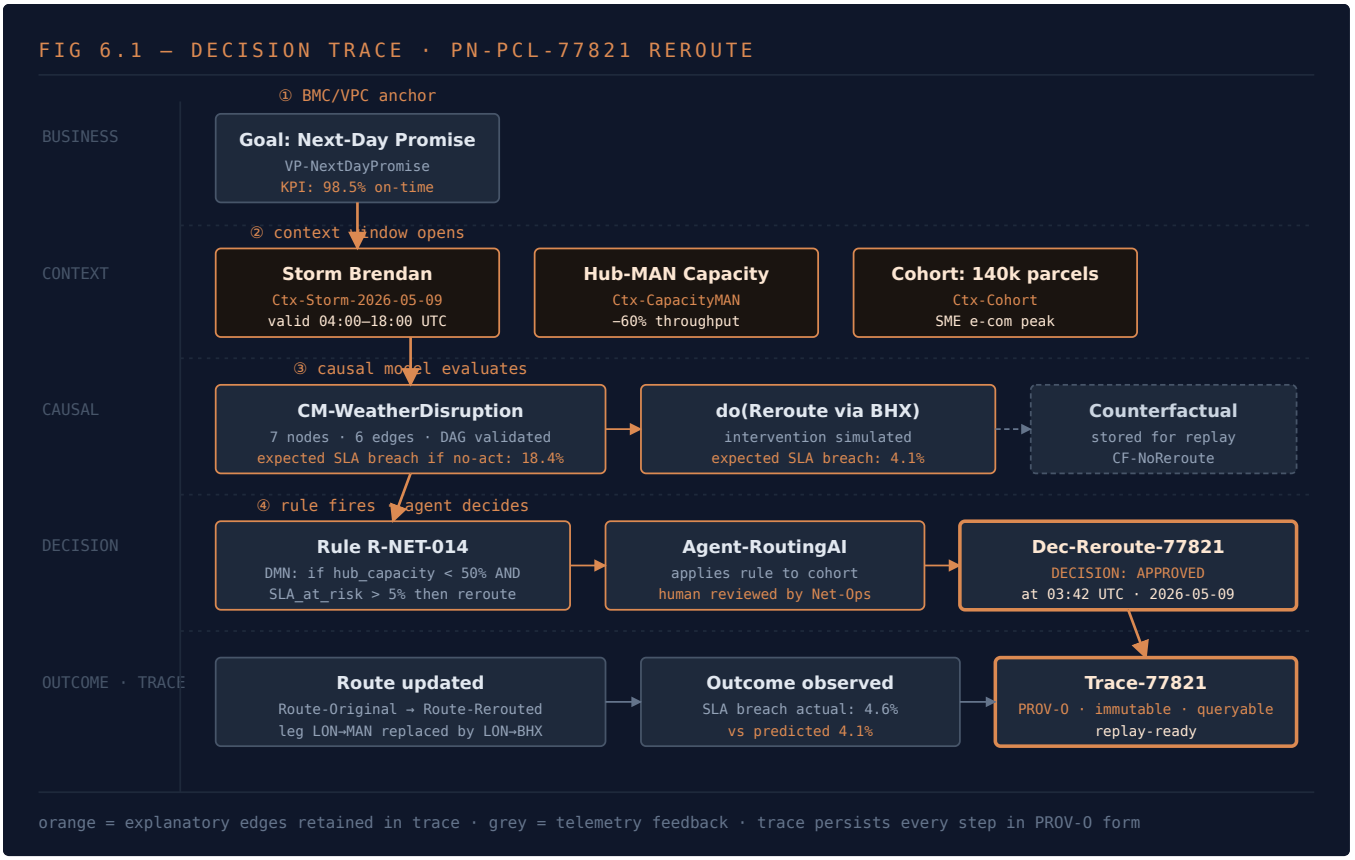


Figure 6.1 — *The end-to-end decision trace. The runtime walks BMC/VPC (anchor: which goal does this decision serve?) → Context (which validity windows are open?) → Causal (what happens if we do nothing vs intervene?) → Decision (which rule fires? which agent acts?) → Outcome (what actually happened?) → Trace (immutable record of all of the above). Each arrow is a stored edge. The trace is itself a graph fragment, queryable by SPARQL.*

§6.2 The trace structure in the graph

The decision and its trace are encoded in the TTL as a connected subgraph. Skeleton (paraphrased from the TTL):

```
pn:Dec-Reroute-77821 a dec:Decision ;
  rdfs:label "Reroute parcel PN-PCL-77821 cohort via Birmingham" ;
  dec:supports pn:VP-NextDayPromise ;
  dec:hasInput pn:Parcel-77821, pn:Route-Original, pn:Ctx-Storm-2026-05-09,
               pn:Ctx-CapacityMAN, pn:Ctx-Cohort ;
  dec:appliesRule pn:R-NET-014 ;
  dec:evaluatedAgainst pn:CM-WeatherDisruption ;
  dec:decidedBy pn:Agent-RoutingAI ;
  dec:reviewedBy pn:Actor-NetOpsDuty ;
  dec:hasOutcome pn:Outcome-Reroute-77821 ;
  dec:hasCounterfactual pn:CF-NoReroute-77821 ;
  dec:hasTrace pn:Trace-77821 ;
  prov:generatedAtTime "2026-05-09T03:42:18Z"^^xsd:dateTime .

pn:Trace-77821 a dec:DecisionTrace ;
  prov:wasDerivedFrom pn:Dec-Reroute-77821 ;
  dec:hasReplayBundle pn:Replay-77821 ;
  dec:hashedAs "sha256:b1e4..." .
```

Every input, every rule, every model, every actor, every outcome and the counterfactual are first-class IRIs, joined to the decision with named predicates. There is no narrative-only field. There is no "comments column."

§6.3 SPARQL on the trace

The pay-off is what becomes queryable. Three example queries.

§6.3.1 — Why was this decision made?

```
PREFIX dec:  <http://postnova.example/ontology/decision#>
PREFIX cau:  <http://postnova.example/ontology/causal#>
PREFIX ctx:  <http://postnova.example/ontology/context#>
PREFIX pn:   <http://postnova.example/instance/>

SELECT ?goal ?contextLabel ?ruleLabel ?agentLabel ?breachIfNoAct ?breachIfReroute
WHERE {
  pn:Dec-Reroute-77821 dec:supports             ?goal ;
                       dec:hasInput             ?ctx ;
                       dec:appliesRule          ?rule ;
                       dec:decidedBy            ?agent ;
                       dec:evaluatedAgainst     ?cm .
  ?ctx   a ctx:Context ;        rdfs:label ?contextLabel .
  ?rule                          rdfs:label ?ruleLabel .
  ?agent                         rdfs:label ?agentLabel .
  ?cm    cau:expectedOutcomeIfNoAction ?breachIfNoAct ;
         cau:expectedOutcomeIfIntervention ?breachIfReroute .
}
```

This query answers *"why this decision?"* in one round trip. It produces six fields that, together, are a complete justification: the goal served, the context inputs, the rule that fired, the agent that decided, the predicted breach if no action, and the predicted breach with intervention.

§6.3.2 — What would have happened without this decision?

```
SELECT ?counterfactualOutcome
WHERE {
  pn:Dec-Reroute-77821 dec:hasCounterfactual ?cf .
  ?cf cau:projectedOutcome ?counterfactualOutcome .
}
```

The counterfactual is stored at decision time, not reconstructed after the fact. This is what enables retrospective audit without re-running models on data that may have drifted.

§6.3.3 — Find every decision in the past 30 days that depended on a closed regulatory context.

```
SELECT ?dec ?ctx ?rule
WHERE {
  ?dec a dec:Decision ;
       dec:hasInput ?ctx ;
       dec:appliesRule ?rule ;
       prov:generatedAtTime ?t .
  ?ctx a ctx:RegulatoryContext ;
       ctx:hasValidity ?v .
  ?v   ctx:validUntil ?expired .
  FILTER (?t > (NOW() - "P30D"^^xsd:duration))
  FILTER (?expired < NOW())
}
```

This is the question regulators and auditors *actually* ask: *"Were there any decisions made on the basis of a regulatory context that has since expired or been revoked?"* In a relational stack, this query takes weeks of

joins across audit logs, regulatory feeds and decision systems. In the graph stack it is one SPARQL query.

The trace is not documentation. The trace is **infrastructure**.

---

§7 — Implementation Roadmap and Maturity Model

A graph-native decision architecture is not built in a sprint. The roadmap below is a five-level maturity model that PostNova (or any equivalent enterprise) can use to assess current state and target future state. Each level is a coherent step — there is no value in pursuing Level 4 capability while Level 2 governance is missing.

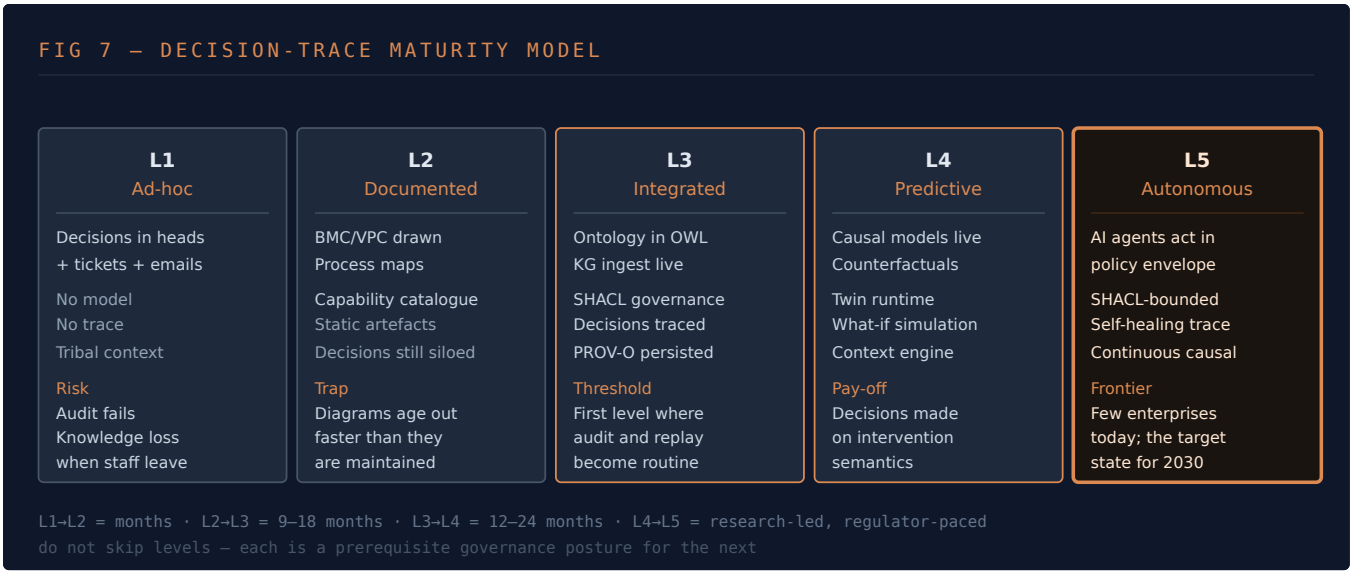


Figure 7 — *Five-level decision-trace maturity model. The interesting threshold is L2 → L3: it is the first level where the architecture stops being a documentation exercise and starts being executable infrastructure. Most enterprises live at L2 and confuse it with L3. The interesting frontier is L4 → L5, where AI agents become legitimate decision-makers within SHACL-bounded policy envelopes — possible only because the trace, the causal model and the context graph give the agents (and their auditors) a shared, queryable reality.*

§7.1 First 90 days

A defensible first 90 days does **not** start with the triplestore. It starts with the ontology TBox and a single decision domain.

- **Days 1–14**: pick one high-stakes, high-frequency decision domain (PostNova: weather-driven rerouting). Identify the BMC blocks it touches, the OMC anchors, the value-stream stages, the involved capabilities, the rules, the actors. Produce the TBox skeleton in OWL.
- **Days 15–30**: stand up Protégé + a triplestore (Apache Jena Fuseki is fine for week one). Load the TBox. Validate with HermiT. Author the SHACL shapes for `dec:Decision`, `cau:CausalEdge`, `ctx:Context`.
- **Days 31–60**: model **one** historical decision end-to-end, by hand if needed, against the TBox. This is the proof point. The decision should be replayable from the trace alone.
- **Days 61–90**: instrument the corresponding production decision service to emit the trace as a graph fragment for every new decision. Set up the SPARQL endpoint behind a GraphQL adapter so consumers can query without writing SPARQL.

The deliverable at day 90 is one decision domain at L3 maturity, with everything else at L2. From there, expansion is by domain, not by component.

§7.2 Anti-patterns of the rollout

- **The Big-Bang Ontology.** Trying to model the whole enterprise before instrumenting any decision. Costs 18 months and produces a static artefact. Always go domain-first.
- **The Replatforming Confusion.** Treating this as a triplestore-procurement project. The architecture is the value; the triplestore is a commodity choice.
- **The Trace-As-Log.** Treating the decision trace as a write-only log. The whole point is that it is queryable structure.

---

§8 — Risks, Anti-patterns and Failure Modes

The architecture has real risks. Naming them earns credibility with sceptics.

§8.1 — Ontology entropy. Without governance, ontologies sprawl. Two teams independently introduce overlapping classes, predicates drift, the SHACL shapes start carving exceptions. Mitigation: a small **ontology council** (3–5 people across architecture, data and business) with weekly cadence and a single-source TBox repository. Bounded contexts (DDD) localise the damage.

§8.2 — Causal model fragility. Causal strengths estimated on stale data give confident wrong answers. Mitigation: causal models carry validity windows like context elements; the twin runtime re-estimates strengths on schedule and emits drift alarms when they shift outside tolerance. The ontology has a `cau:lastValidated` predicate that SHACL can enforce.

§8.3 — Context overload. A naïve context engine retrieves "all of context" for every decision. The graph collapses under retrieval cost. Mitigation: context elements bind to OMC anchors and value-stream stages; the engine only retrieves context whose binding intersects the affected anchors and stages.

§8.4 — Vector-RAG temptation. A new team joins and proposes "let's just embed the whole ontology and do similarity search". This is a category error, but a tempting one because vector RAG is faster to demo. Mitigation: keep vector retrieval as a **lookup channel into the graph**, not a substitute for the graph. The vector index returns IRIs; the IRIs are walked in the graph. Every retrieval is a graph traversal at heart.

§8.5 — Performance ceilings on monolithic graphs. A single triplestore holding billions of triples is a single point of failure and a query-planner nightmare. Mitigation: bounded contexts as named graphs, SPARQL federation across domain stores, and a query gateway that knows the domain map. This is also why Data Mesh aligns so well — it pre-distributes the load.

§8.6 — Trace inflation. Persisting full trace fragments for every micro-decision can balloon storage. Mitigation: tiered persistence — hot store for last 90 days, warm store for one year, cold archive thereafter. The trace ontology supports `dec:traceTier` natively. Critical decisions (regulatory, financial, customer-facing) override the tier policy and stay hot.

§8.7 — The skill cliff. SPARQL, OWL, SHACL, Pearl-style causal modelling and DMN are *each* niche. Hiring a team that has all five is hard. Mitigation: split the skills across roles (semantic engineer, causal scientist, decision designer, data-product owner) and converge them through the ontology council. Do not look for a unicorn.

---

§9 — How the Expert Sees It Differently

Most architects who encounter this stack interpret it as *"better enterprise architecture documentation"*. That reading is wrong. It is the reading that keeps enterprises stuck at L2 maturity, mistaking pretty diagrams for executable infrastructure. The expert reading runs deeper.

§9.1 — The graph is the system, not a model of the system. When the BMC is a graph, when the value stream is a graph, when the decision and its trace are graph fragments persisted in the same store, the architecture stops being a *representation* of the enterprise and starts being part of its *operational fabric*. The ArchiMate diagram is now a query result, not a hand-drawn slide. The capability heatmap is a SPARQL aggregation, not a workshop output. The lineage column in a regulator's submission is a path traversal, not a one-off project.

§9.2 — Decisions, not data, are the atomic unit. The dominant data narrative of the past decade has been *"data is the new oil"*. The graph stack rejects this. Data is fuel; *decisions* are the atomic unit of enterprise value. An architecture that does not have a first-class `dec:Decision` class is treating its primary subject matter as a side-effect of its data products. Reverse the polarity: build the architecture around the decision, and the data products fall out as inputs.

§9.3 — Causal goes before correlational. The hardest cultural shift is moving the analytics function from correlation (*"customers who churned tended to have more delivery exceptions"*) to causation (*"a 1% increase in next-day SLA breach causes a 0.6% increase in 90-day churn, controlling for cohort and tenure"*). The first sentence is BI. The second is decision-grade. Pearl's framework is not optional; it is the only formalism that lets enterprises survive the regulatory turn on AI accountability.

§9.4 — Context is a separate layer, not a payload field. Architects who try to encode context as fields on existing entities (`order.weather_at_time_of_decision = "stormy"`) end up with denormalised, drift-prone records that lose the *binding* and *validity* that makes context useful. Context is a layer with its own lifecycle, its own validity windows, its own bindings to OMC anchors. Treat it as such, or surrender any hope of replayability.

§9.5 — Twins are decision proxies, not visualisations. The current digital-twin literature is full of dashboards with rotating 3D models. The expert view: a digital twin's purpose is to be the addressable object that the decision service queries, with state and a linked causal model that can be intervened upon. If the twin cannot answer *"if I do X, what happens?"* in causal-model semantics, it is a dashboard wearing the word *twin*. The PostNova capability twin in §5.6 — linked to `CM-WeatherDisruption`, simulator-bound, decision-emitting — is the legitimate form.

§9.6 — The investment thesis is on the decision trace, not the ontology. Enterprises that buy the ontology pitch and not the trace pitch end up at L2. The trillion-dollar-class observation in the recent context-graph literature (Foundation Capital, 2025) is not that ontologies are valuable; it is that *the missing infrastructure layer is the one that makes context retrievable, decisions traceable and causes intervenable*. The ontology is a means; the trace is the end. PhD researchers writing on this should resist the temptation to treat the ontology as the contribution; the contribution is the trace as governable, replayable, queryable infrastructure.

§9.7 — The 0.1% framing. Most architects, asked to design this stack, will produce something that looks plausible and yet stops at the first hard problem: how do you keep the causal strengths fresh? How do you bound a context window? How do you stop the SHACL shapes from accumulating exceptions until they are vacuous? The 0.1% architects build governance into the ontology itself — `cau:lastValidated`, `ctx:hasValidity`, `dec:traceTier`, `sh:closed true` on critical shapes — so the system polices itself instead of relying on a runbook. The PhD-grade contribution is to formalise *which* governance predicates the

ontology must carry to be self-policing, and to prove that the resulting architecture is **survivable** at enterprise scale across decade-length time horizons. That is the open research frontier this whitepaper points toward.

---

§A — Appendix: SPARQL Query Library

A minimum-viable query library for working with the PostNova graph. All queries assume the prefixes from §6.3.

§A.1 Decision-trace queries

A.1.1 — Full provenance of a decision

```
SELECT ?p ?o
WHERE { pn:Dec-Reroute-77821 ?p ?o }
```

A.1.2 — All decisions made by a specific agent in the past 24 hours

```
SELECT ?dec ?t WHERE {
  ?dec a dec:Decision ;
       dec:decidedBy pn:Agent-RoutingAI ;
       prov:generatedAtTime ?t .
  FILTER (?t > (NOW() - "PT24H"^^xsd:duration))
} ORDER BY DESC(?t)
```

A.1.3 — Decisions whose outcome diverged > 5% from causal-model prediction

```
SELECT ?dec ?predicted ?actual WHERE {
  ?dec a dec:Decision ;
       dec:evaluatedAgainst ?cm ;
       dec:hasOutcome ?out .
  ?cm  cau:expectedOutcomeIfIntervention ?predicted .
  ?out cau:observedOutcome ?actual .
  FILTER (ABS(?actual - ?predicted) > 0.05)
}
```

A.1.3 is the model-drift alarm: a model whose decisions repeatedly diverge from prediction is a model that needs re-estimation.

§A.2 Causal queries

A.2.1 — Full DAG of a causal model

```
SELECT ?from ?to ?strength WHERE {
  pn:CM-WeatherDisruption cau:hasEdge ?e .
  ?e cau:from ?from ; cau:to ?to ; cau:strength ?strength .
}
```

A.2.2 — Find all decisions that depend on the same root cause

```
SELECT DISTINCT ?dec WHERE {
  ?dec dec:evaluatedAgainst ?cm .
  ?cm  cau:hasNode pn:Node-Storm .
}
```

§A.3 Context queries

A.3.1 — Currently valid context elements bound to a hub location

```
SELECT ?ctx ?label WHERE {
  ?ctx a ctx:Context ;
       rdfs:label ?label ;
       ctx:bindsTo pn:Hub-MAN ;
       ctx:hasValidity ?v .
  ?v ctx:validFrom ?start ; ctx:validUntil ?end .
  FILTER (?start <= NOW() && NOW() < ?end)
}
```

A.3.2 — Decisions made on the basis of expired context (regulator's question)

```
SELECT ?dec ?ctx WHERE {
  ?dec a dec:Decision ;
       dec:hasInput ?ctx ;
       prov:generatedAtTime ?dt .
  ?ctx ctx:hasValidity ?v .
  ?v   ctx:validUntil ?expired .
  FILTER (?expired < ?dt)
}
```

A.3.2 is the question that defines a regulatory-grade trace store. If your audit infrastructure cannot answer it in one query, you do not have one.

§A.4 BMC + capability queries

A.4.1 — Which value propositions are at risk if Hub-MAN is offline?

```
SELECT DISTINCT ?vp WHERE {
  pn:Hub-MAN cap:supports ?cap .
  ?cap proc:realizedBy ?proc .
  ?proc proc:partOf ?stage .
  ?stage proc:partOf pn:VS-OrderToDoorstep .
  pn:VS-OrderToDoorstep bmc:supports ?vp .
}
```

This is the strategic-impact query: a single SPARQL traversal from a physical location to the value propositions that depend on it.

---

§B — Appendix: SHACL Validation Patterns

The canonical shapes that govern the graph. Excerpted from the TTL.

```
pn:DecisionTraceShape a sh:NodeShape ;
  sh:targetClass dec:Decision ;
  sh:property [ sh:path dec:hasInput      ; sh:minCount 1 ] ;
  sh:property [ sh:path dec:appliesRule   ; sh:minCount 1 ] ;
  sh:property [ sh:path dec:hasOutcome    ; sh:minCount 1 ] ;
  sh:property [ sh:path dec:supports      ; sh:minCount 1 ;
                sh:class lead:Goal ] ;
  sh:property [ sh:path dec:hasTrace      ; sh:minCount 1 ] ;
  sh:property [ sh:path dec:evaluatedAgainst ;
                sh:minCount 1 ; sh:class cau:CausalModel ] ;
  sh:closed true ;
  sh:ignoredProperties ( rdf:type rdfs:label rdfs:comment ) .

pn:CausalEdgeShape a sh:NodeShape ;
  sh:targetClass cau:CausalEdge ;
  sh:property [ sh:path cau:from     ; sh:minCount 1 ; sh:class cau:CausalNode ] ;
  sh:property [ sh:path cau:to       ; sh:minCount 1 ; sh:class cau:CausalNode ] ;
  sh:property [ sh:path cau:strength ; sh:minCount 1 ;
                sh:datatype xsd:decimal ;
                sh:minInclusive -1.0 ;
                sh:maxInclusive  1.0 ] .

pn:ContextValidityShape a sh:NodeShape ;
  sh:targetClass ctx:Context ;
  sh:property [ sh:path ctx:hasValidity ;
                sh:minCount 1 ;
                sh:class ctx:Validity ] ;
  sh:property [ sh:path ctx:bindsTo ;
                sh:minCount 1 ] .
```

The crucial design decision is `sh:closed true` on `DecisionTraceShape`. Closed shapes prevent silent property drift: if a team starts adding novel predicates to decisions, validation fails and the ontology council is notified. This is governance through executable constraint, not policy document.

---

§C — Appendix: Glossary

- **Bounded Context (DDD)** — a domain boundary within which a model's terms are canonical; encoded as a named graph in RDF.
- **BMC** — Business Model Canvas (Osterwalder & Pigneur). Nine-block strategic canvas; here, an ontology TBox layer.
- **Capability (BIZBOK)** — what the enterprise *does*, distinct from how (process). Persistent across reorgs.
- **Capability Twin** — a digital twin scoped to a capability rather than an asset; carries state, telemetry, simulator and a linked causal model.
- **Causal Graph** — a directed acyclic graph (DAG) whose edges represent causal-not-correlational dependence (Pearl, 2009).
- **Context Graph** — a graph layer holding situational facts (weather, capacity, regulatory status) with validity windows and bindings.
- **Counterfactual** — projected outcome under an alternative intervention, computed at decision time and stored in the trace.
- **Decision Service** — application-layer service that evaluates rules / models, persists the trace, returns the outcome.
- **Decision Trace** — graph fragment recording a decision's inputs, rule, model, agent, outcome and counterfactual; PROV-O-aligned.

- **DMN** — Decision Model and Notation (OMG). Standard for decision tables and FEEL expressions.
- **Folksonomy** — emergent, user-driven tagging vocabulary; opposite of formal ontology.
- **Knowledge Graph** — a graph of typed entities and relations grounded in an ontology, populated with instance data.
- **OMC** — Operating Model Canvas (Campbell). Six-block POLISM framework: Processes, Organisation, Locations, Information, Suppliers, Management Systems.
- **Ontology (TBox)** — formal specification of conceptualisation: classes, relations, axioms (Gruber, 1993; Studer et al., 1998).
- **PROV-O** — W3C provenance ontology; standard vocabulary for data lineage.
- **SHACL** — Shapes Constraint Language (W3C). Constrains the shape of an RDF graph; the validation layer.
- **TBox / ABox** — terminology box (schema, classes) / assertion box (instances, facts).
- **VPC** — Value Proposition Canvas (Osterwalder). Pains/Gains/Jobs ↔ Pain Relievers/Gain Creators/Products & Services.

---

§D — References

- BIZBOK Guide. *A Guide to the Business Architecture Body of Knowledge.* Business Architecture Guild.
- Campbell, A., Gutierrez, M., & Lancelott, M. *Operating Model Canvas.* Van Haren Publishing.
- Dehghani, Z. (2022). *Data Mesh: Delivering Data-Driven Value at Scale.* O'Reilly.
- Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison-Wesley.
- Foundation Capital (2025). *Context graphs: AI's trillion-dollar opportunity.* Foundation Capital perspectives.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199–220.
- Guarino, N. (1998). Formal ontology and information systems. *FOIS '98.*
- Modern Data 101 (2025). *AI-Ready Data vs Analytics-Ready Data.* Modern Data 101 perspectives.
- Object Management Group. *Decision Model and Notation (DMN) 1.4.*
- Osterwalder, A., & Pigneur, Y. (2010). *Business Model Generation.* Wiley.
- Osterwalder, A., Pigneur, Y., Bernarda, G., & Smith, A. (2014). *Value Proposition Design.* Wiley.
- Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* (2nd ed.). Cambridge University Press.
- Pearl, J., & Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect.* Basic Books.
- SAP. *S/4HANA Industry Cloud — Postal & Logistics Reference Models.* SAP industry documentation.
- Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1–2), 161–197.
- The Open Group. *TOGAF Standard, Version 9.2.* Architecture Development Method.
- The Open Group. *ArchiMate 3.2 Specification.*
- W3C. *RDF 1.2 Concepts and Abstract Syntax.* W3C Recommendation.
- W3C. *OWL 2 Web Ontology Language: Structural Specification.* W3C Recommendation.
- W3C. *Shapes Constraint Language (SHACL).* W3C Recommendation.
- W3C. *PROV-O: The PROV Ontology.* W3C Recommendation.
- W3C. *SPARQL 1.1 Query Language.* W3C Recommendation.

- APQC. *Process Classification Framework (PCF) — Cross-Industry, Version 7.4.* APQC.

End of whitepaper. Companion artefacts: `postnova-ontology.ttl` *(Protégé-loadable enterprise ontology) and* `README.md` *(environment setup and reproduction instructions).*