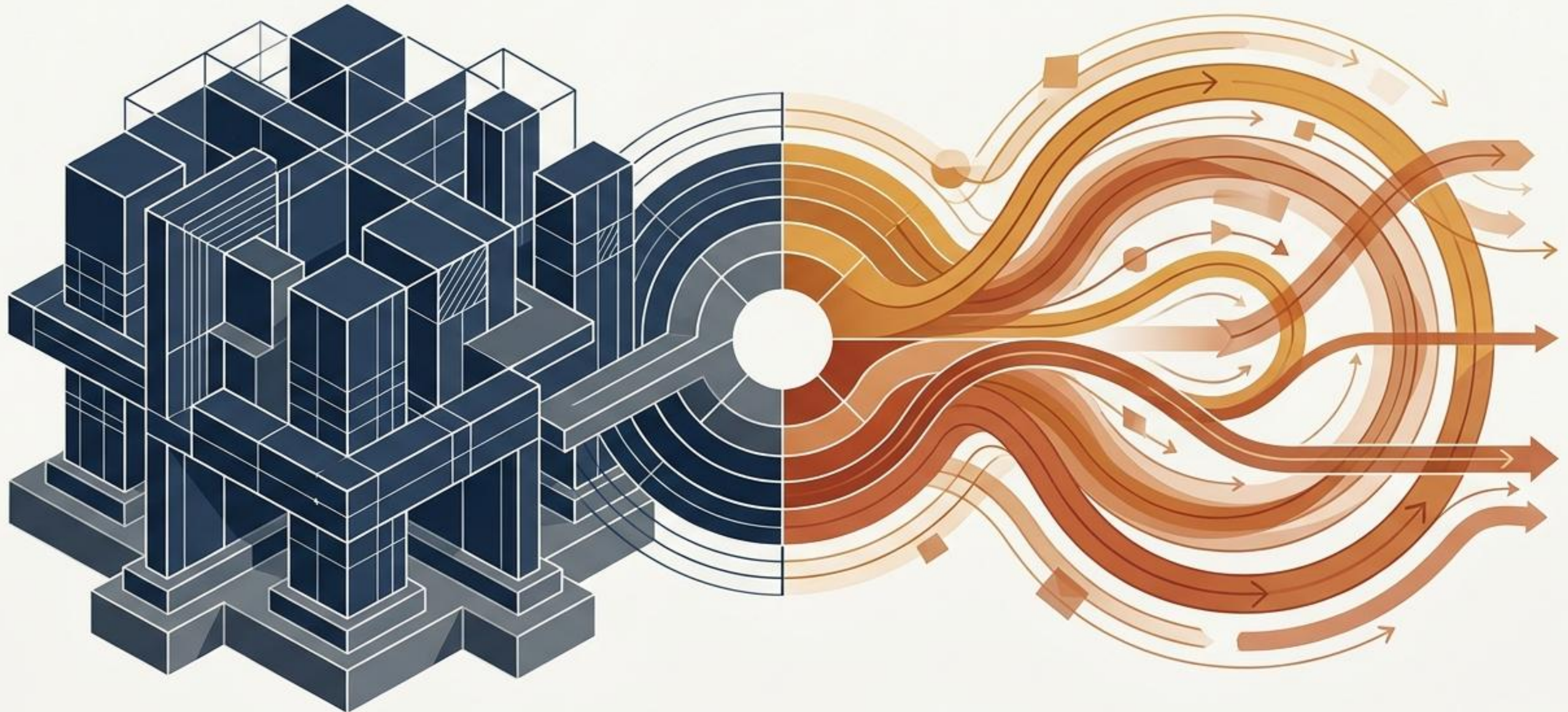
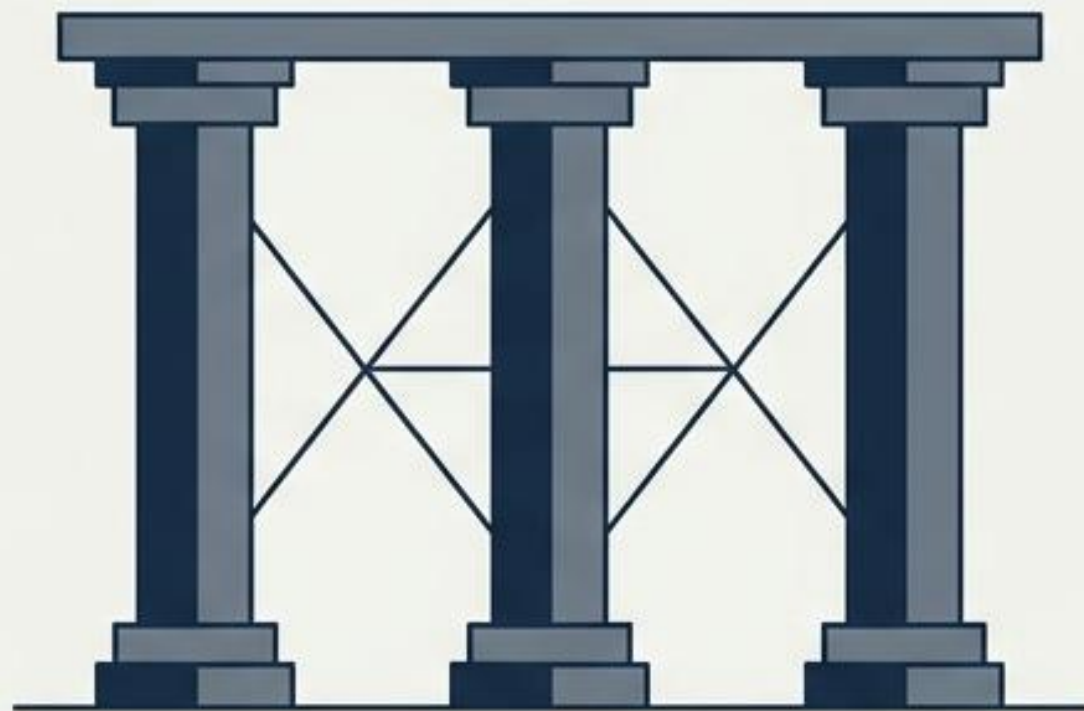


THE UNIFIED INSURANCE ONTOLOGY

Integrating Data Knowledge and Data Action



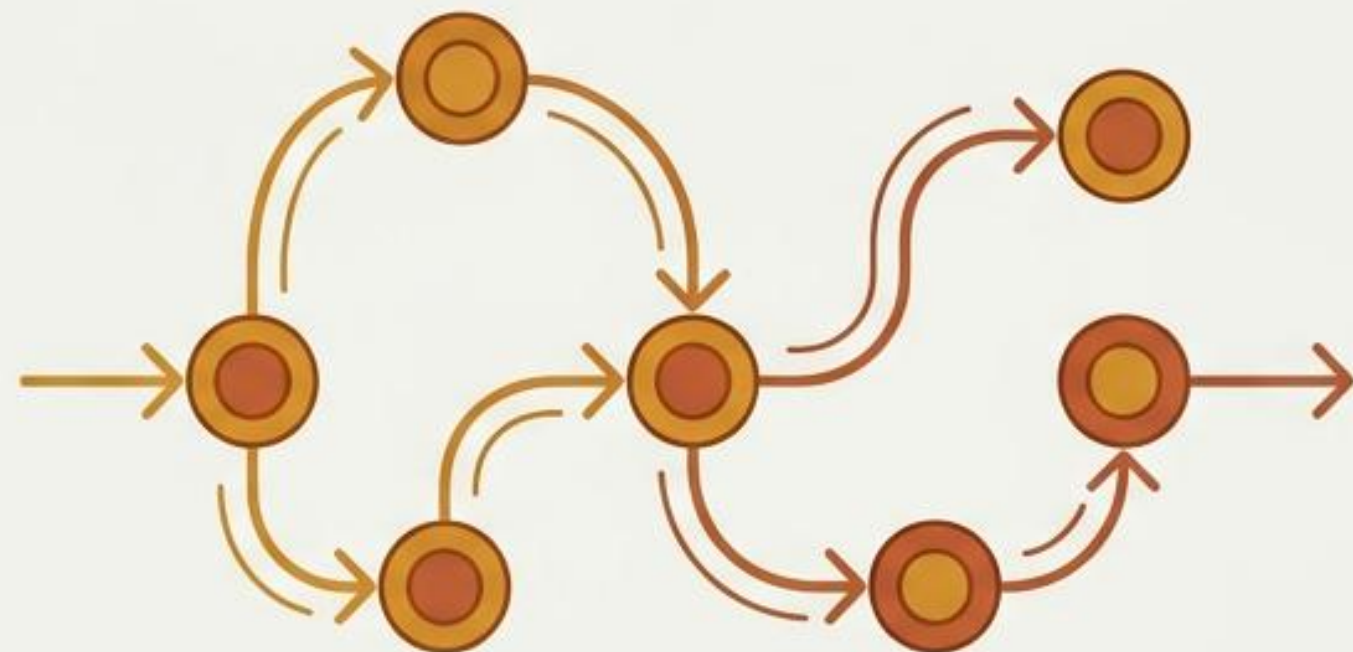
Data at Rest: The Knowledge Perspective



What does the enterprise know?

- Policies
- Coverages
- Claims
- Premiums

Data in Motion: The Action Perspective



What is the enterprise doing?

- First Notice of Loss submitted
- Quote approved
- Policy bound

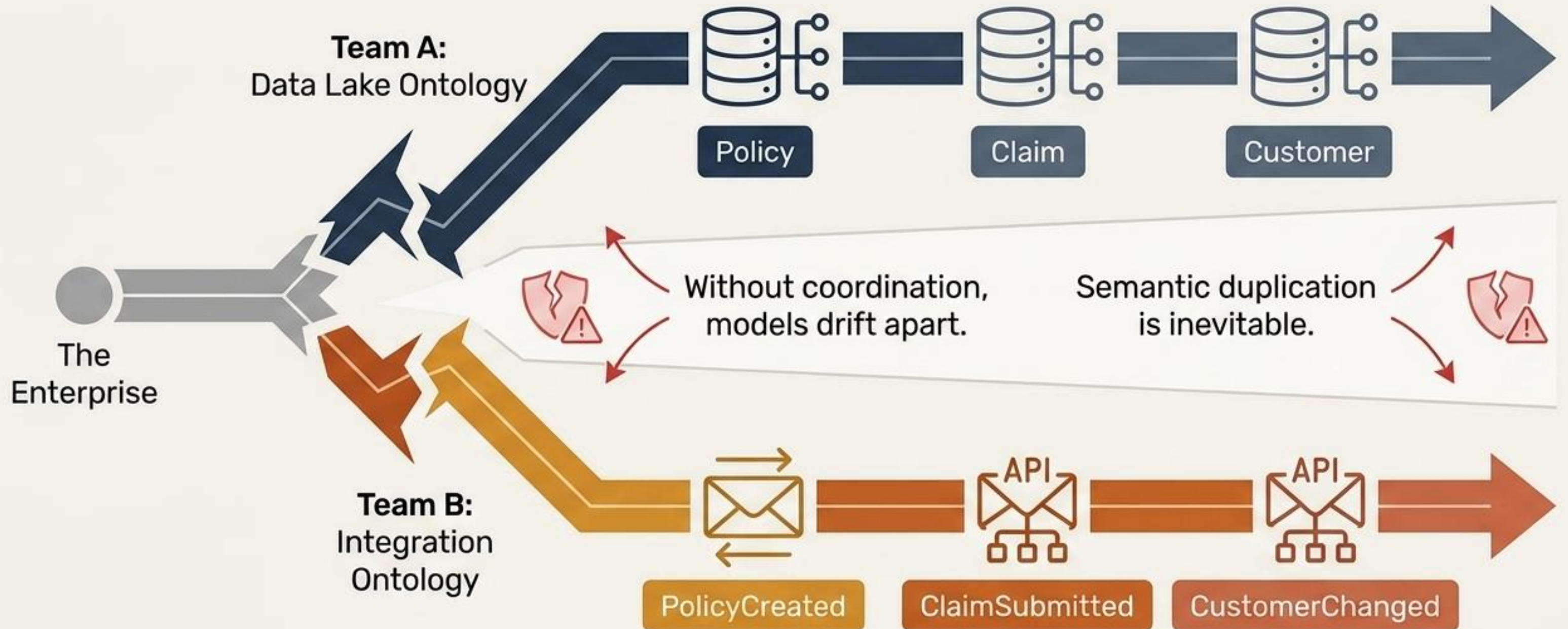
The Anatomy of Rest vs. Motion

Dimension	Data at Rest (Knowledge)	Data in Motion (Action)
The Nature	Enduring Facts	Interactions and Changes
The Habitat	Data Lakes, Master Data Repositories, Data Warehouses	APIs, Kafka Topics, Event Streams, EDI
The Examples	Policy Status, Coverage Limits, Named Insured	Quote Requested, Claim Assigned, Fraud Investigation Triggered

The Semantic Blind Spot in Modern Architecture



The Anti-Pattern: Modeling the Enterprise in Silos



The Hidden Costs of Semantic Divergence



Multiple Definitions

Overlapping concepts fracture understanding:
Customer vs. Policyholder vs. Named Insured
vs. Account Holder.



Inconsistent Identifiers

System A uses policyId, System B uses
policyNumber, System C uses contractReference
without clear equivalence.



Duplicate Mappings

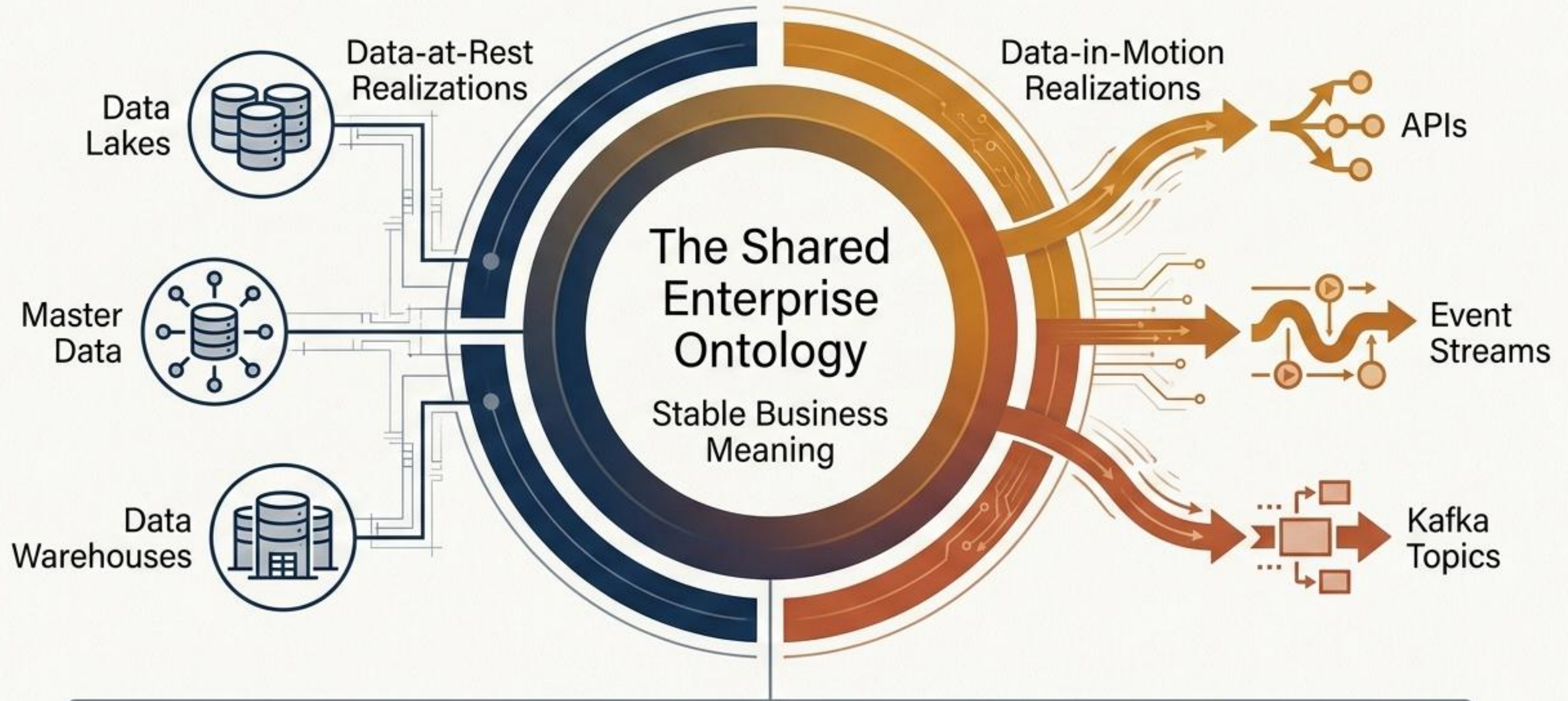
Custom translations are built for every integration,
leading to massive accumulations of Semantic Debt.



Broken Lineage

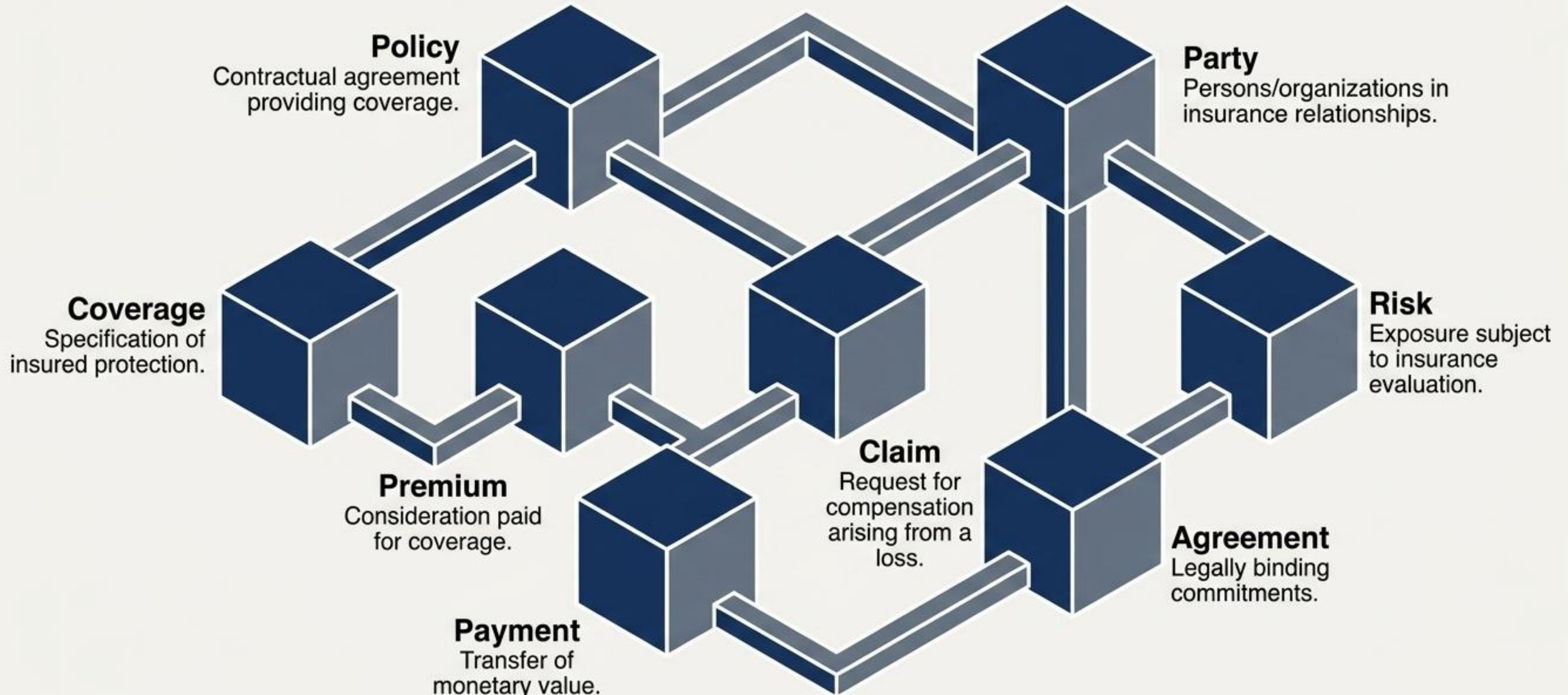
Analysts cannot answer: Which event caused this
data? Which transaction modified this record?

The Solution: A Shared Semantic Core

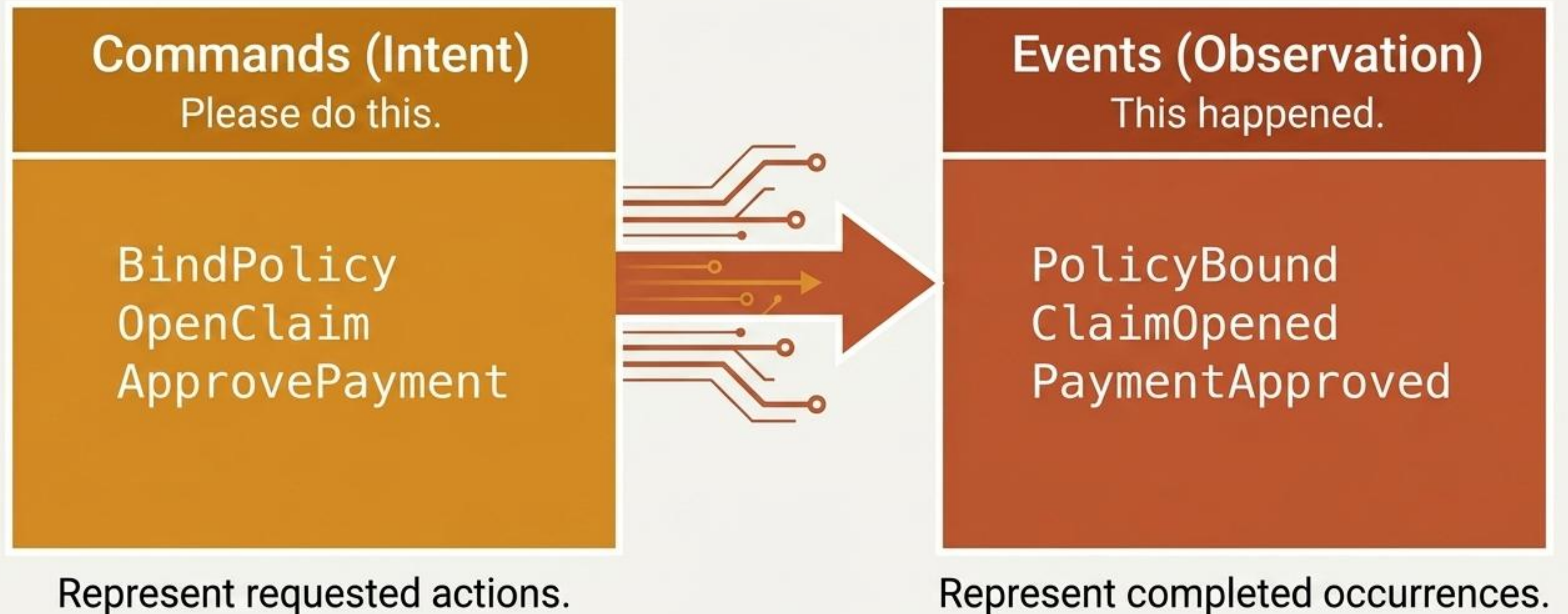


Preserve consistency at the core while allowing technical specialization at the edge.

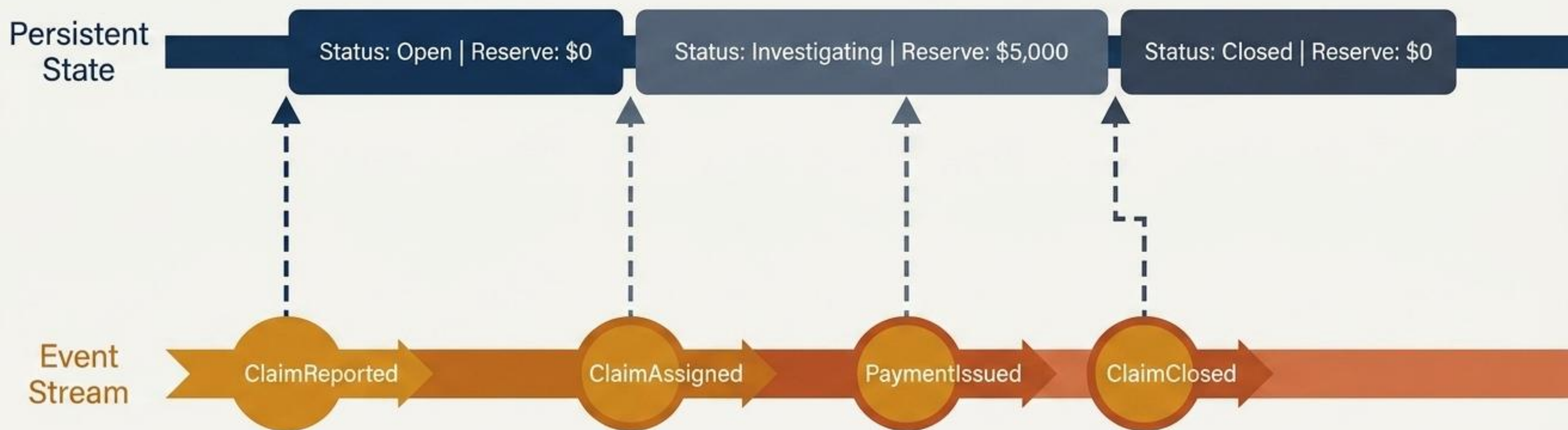
The Foundation: Enterprise Core Concepts



The Dynamic Layer: Distinguishing Intent from Observation



State Transitions in Practice: The Claim Lifecycle



Persistent records capture current state. Events explain how the state emerged.

APIs as Semantic Contracts: Structure vs. Meaning

Structure

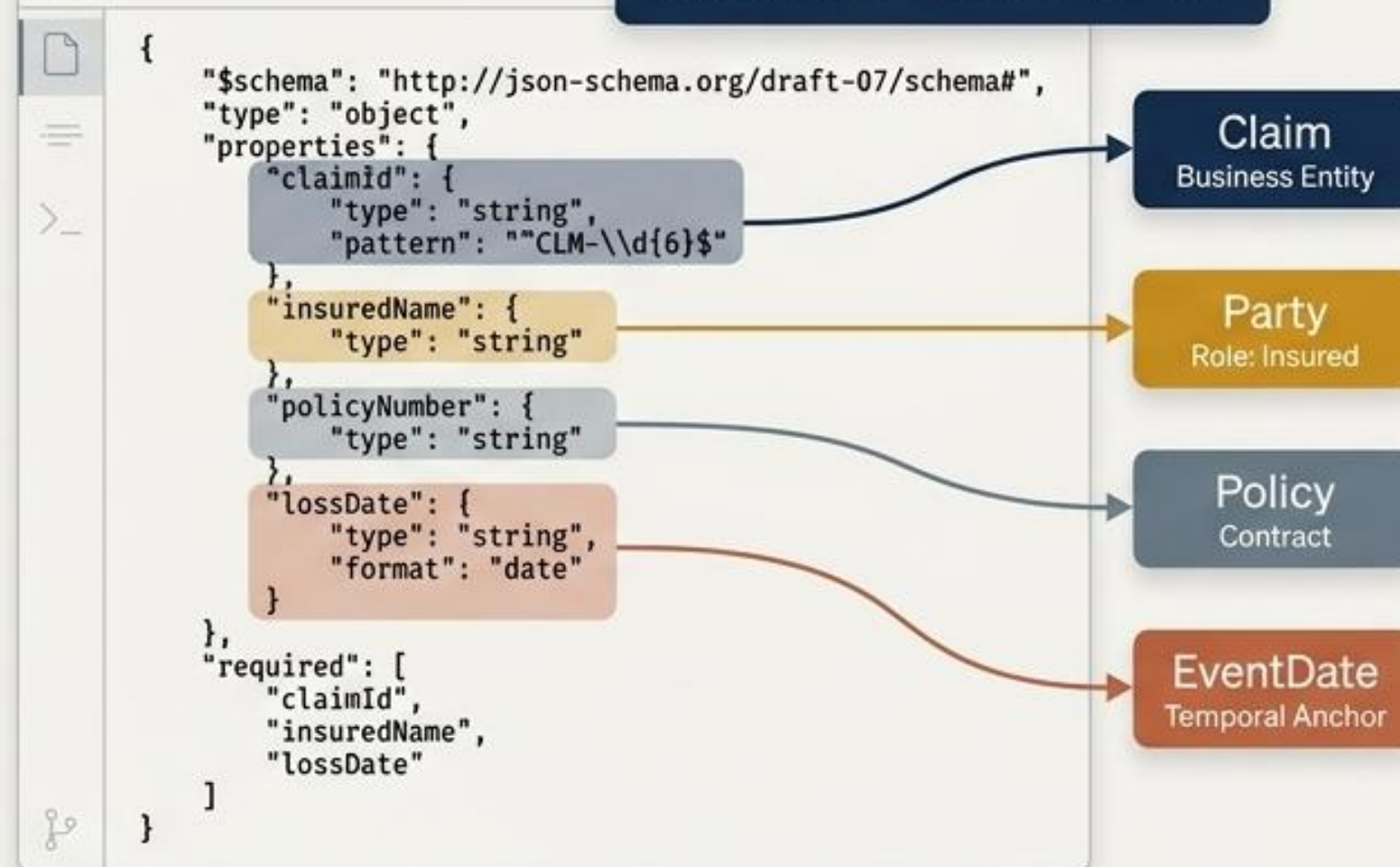
Is this payload valid?

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "claimId": {
      "type": "string",
      "pattern": "CLM-\\d{6}$"
    },
    "insuredName": {
      "type": "string"
    },
    "policyNumber": {
      "type": "string"
    },
    "lossDate": {
      "type": "string",
      "format": "date"
    }
  },
  "required": [
    "claimId",
    "insuredName",
    "lossDate"
  ]
}
```

Checks data types, syntax, and required fields.

Meaning

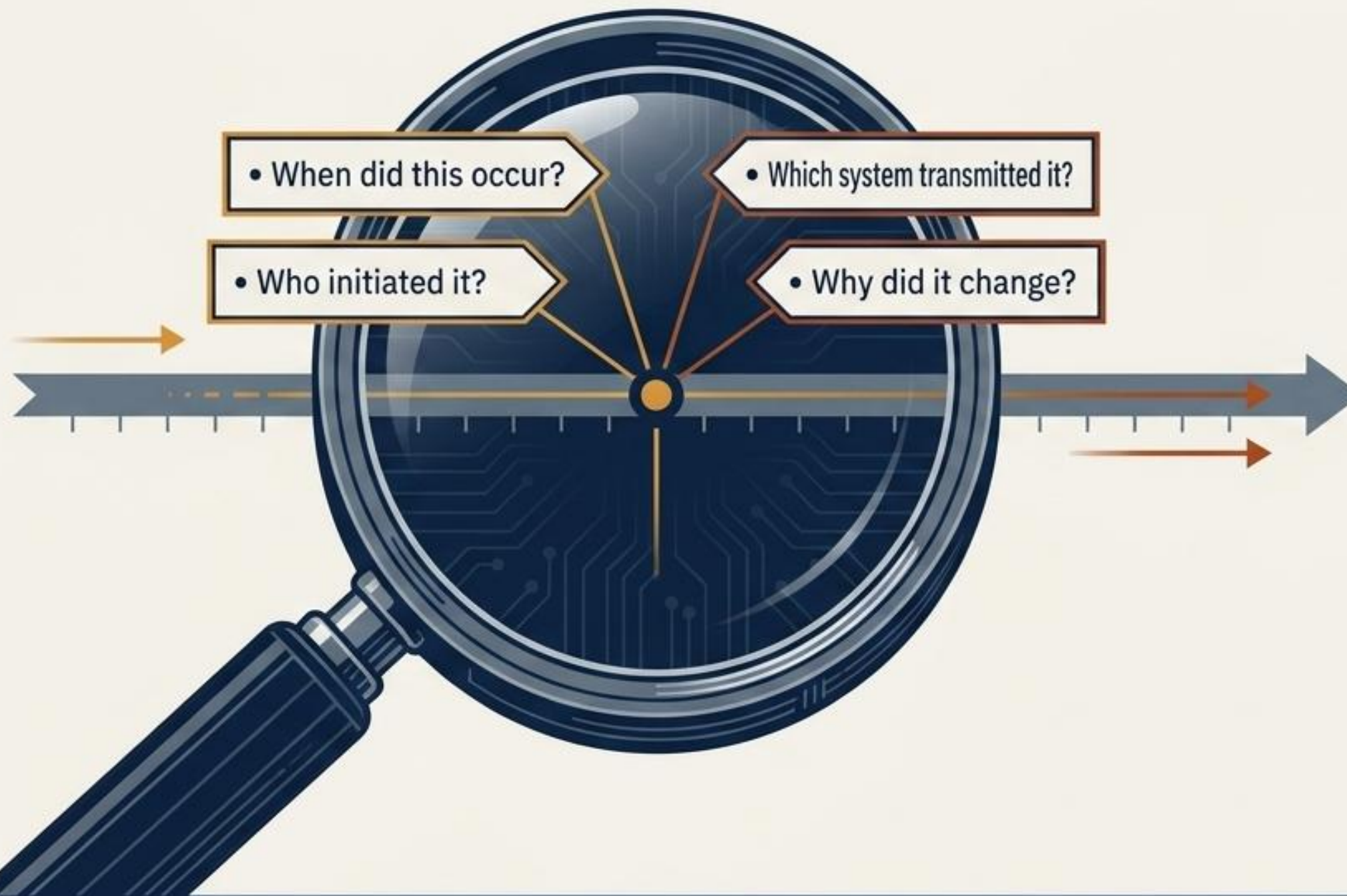
What does this mean?



Identifies: What is a claim? Who is the insured?

JSON schemas define **structure**. Ontologies define meaning. Both are absolutely necessary.

Unlocking Provenance and Temporal Semantics

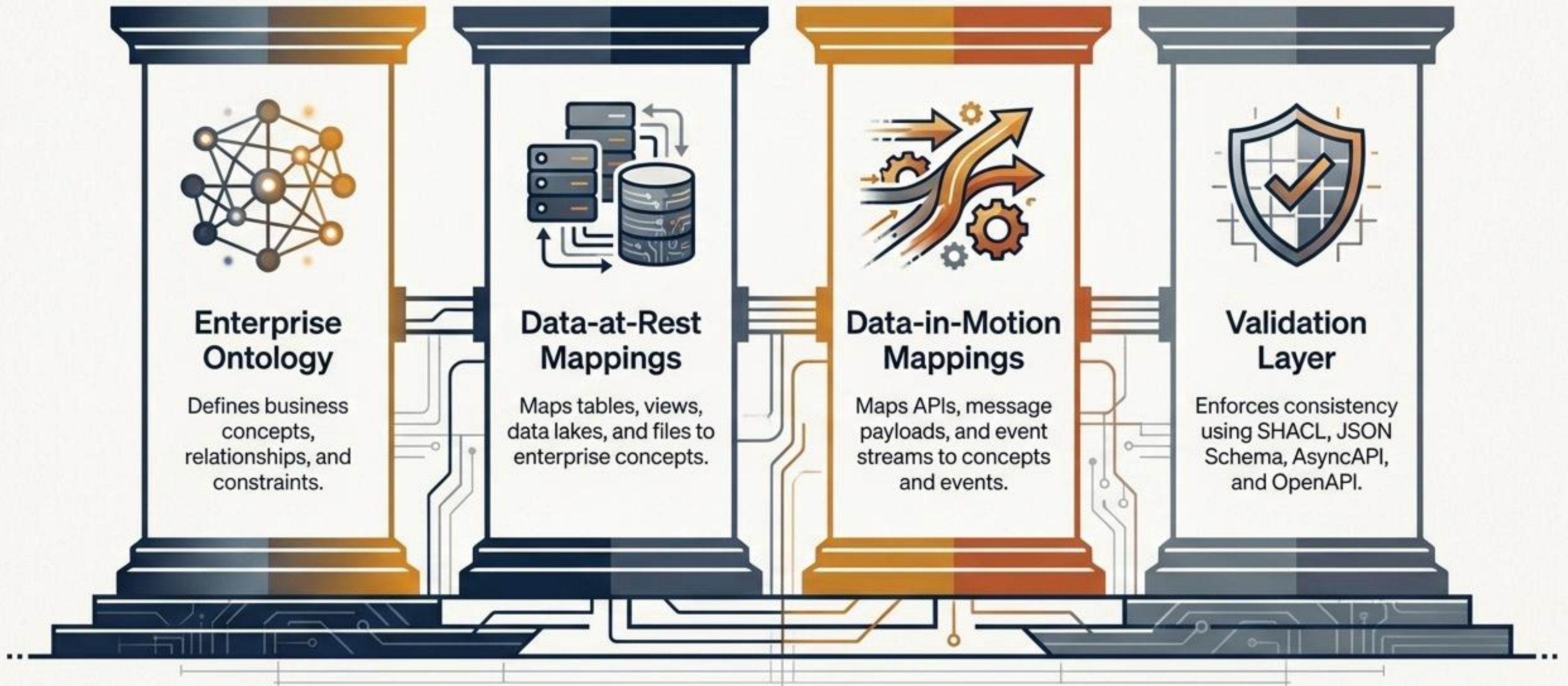


Enterprise Capabilities Enabled:

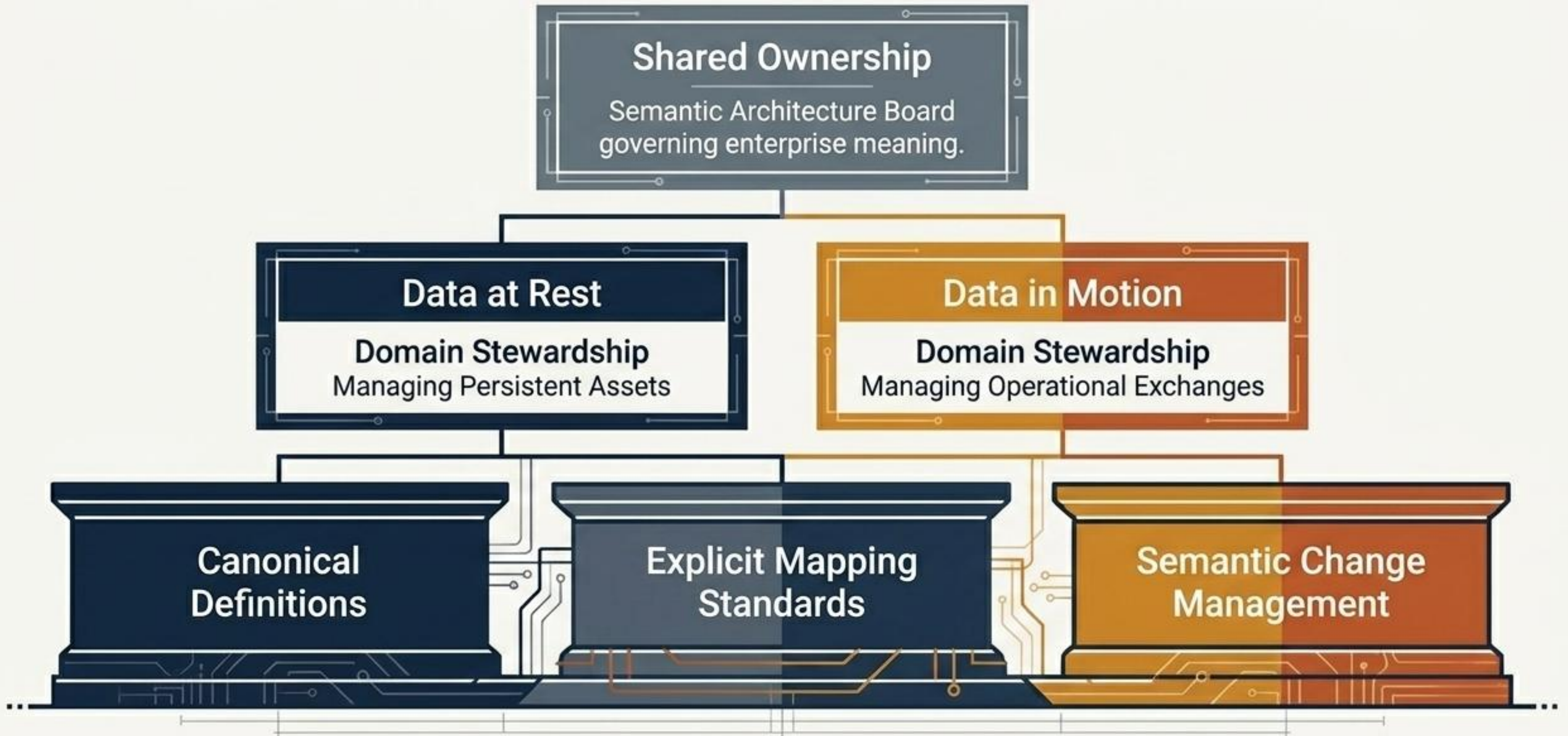
- ✓ Auditability
- ✓ Regulatory Compliance
- ✓ Explainability
- ✓ Root-cause analysis

Persistent data alone cannot answer these questions.

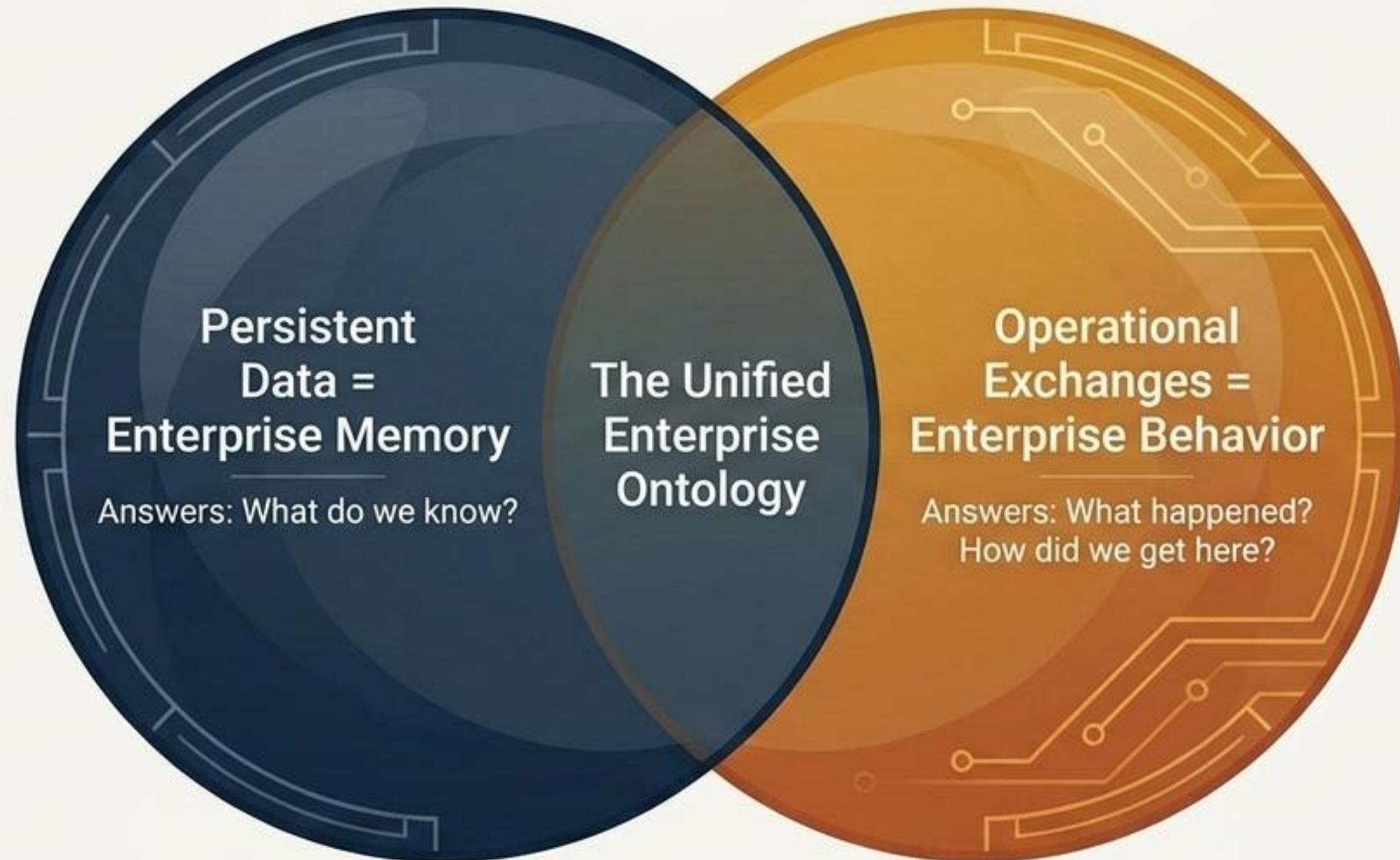
The Implementation Blueprint: A Four-Pillar Mapping Strategy



Governing the Unified Enterprise Ontology



The Synthesis of Enterprise Memory and Behavior



Only by integrating both perspectives can the organization accurately answer:
What should we do next?