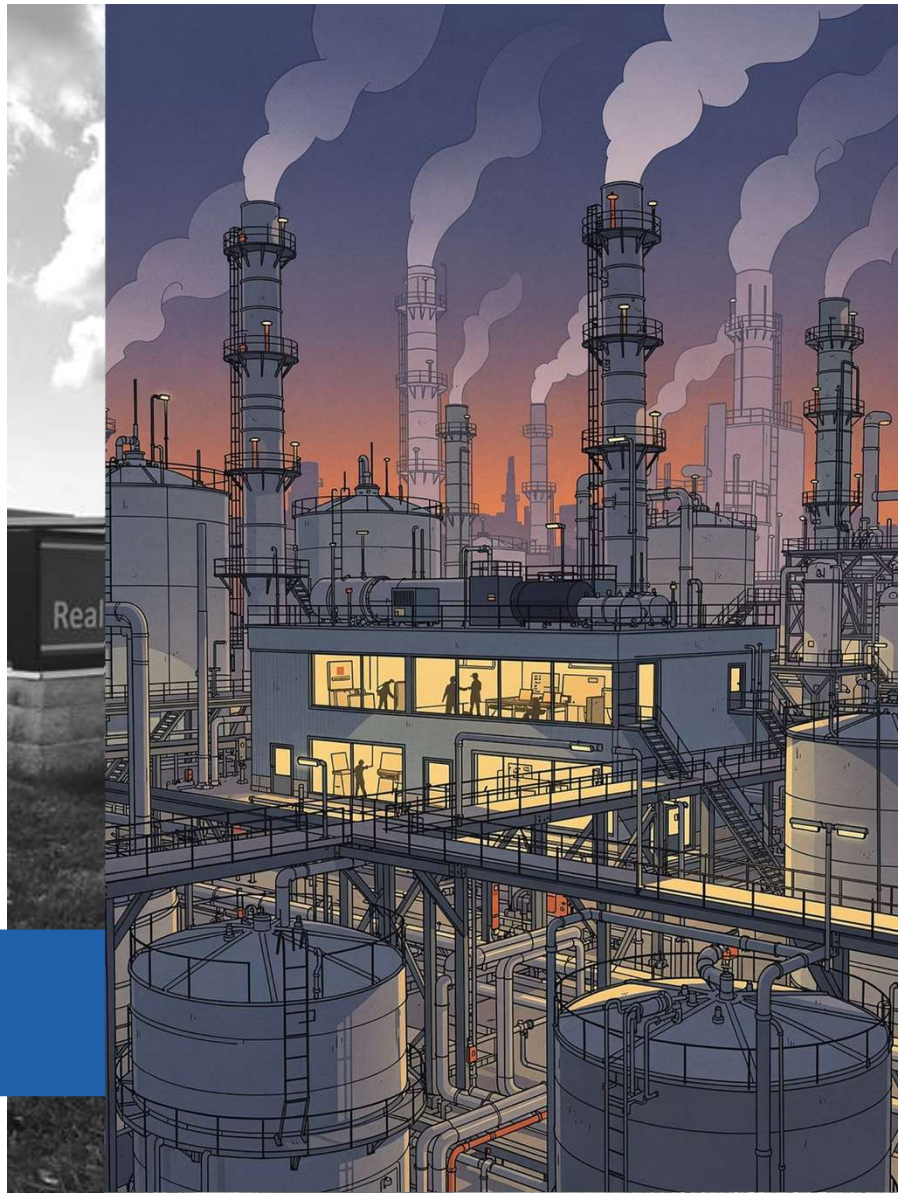
REAL TIME AUTOMATION®

Why Digital Transformation Often Dies a Slow Death on the Factory Floor

ICHEME PMC SIG WEBINAR — 15 JULY 2026

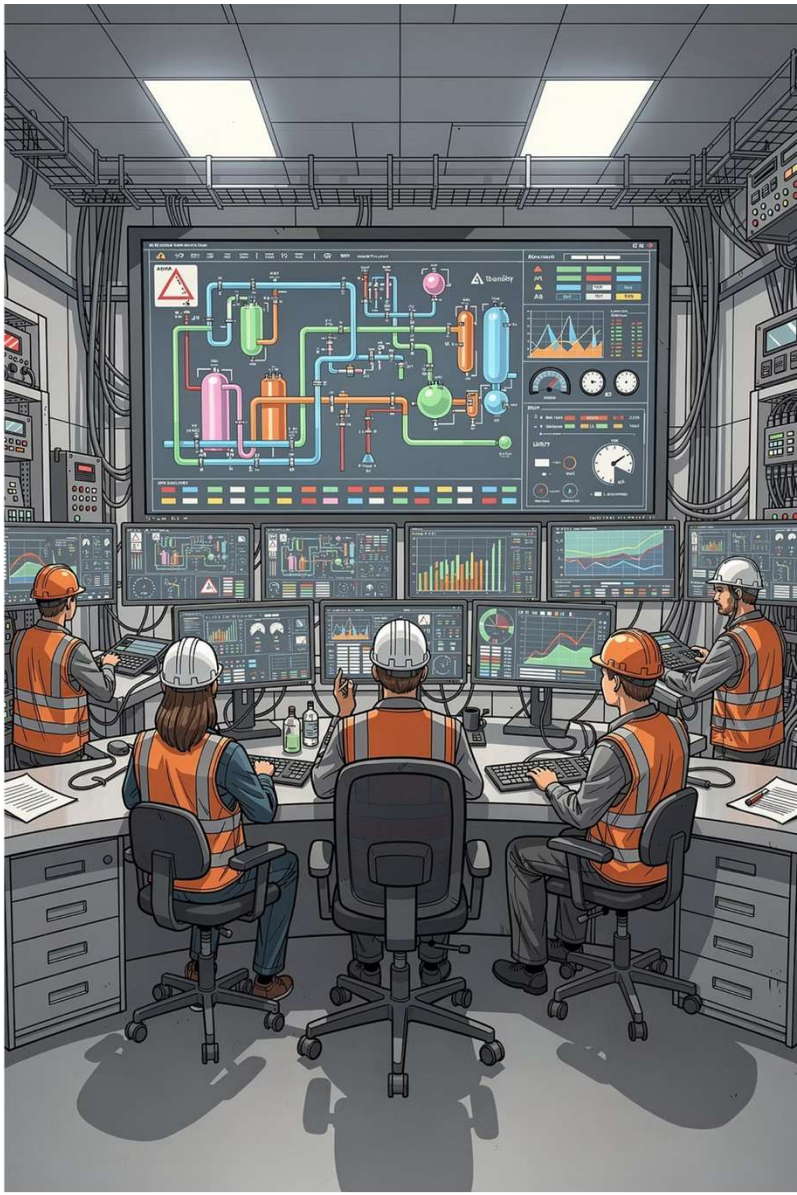


A Practical Blueprint For AI-ready Manufacturing Data

Part 1

The Problem

The Slow Death



JOHN RINALDI – REAL TIME AUTOMATION

"The Digital Transformation is Unrealistic
For Many Manufacturers as There is
Generally, Too Much Chaos On Their Plant Floor."

- NO Asset Lists
- NO Segmentation
- NO Data Modeling
- NO Cybersecurity

- Author of six industrial automation books

  Industrial Ethernet, OPC UA, Modbus, EtherNet/IP

- Founder, Real Time Automation (RTA)

  Pewaukee, WI — serving plant engineers worldwide

- Recognized OT & cybersecurity expert

  Thousands of articles on industrial networking and automation strategy

JOHN RINALDI – REAL TIME AUTOMATION

"The Digital Transformation is Unrealistic
For Many Manufacturers as There is
Generally, Too Much Chaos On Their Plant Floor."

- NO Asset Lists
- NO Segmentation
- NO Data Modeling
- NO Cybersecurity

Today's promise:

10 Actionable Strategies To Turn Chaotic Plant Data Into AI-ready Information.

The Promise vs. The Reality

The Pitch	The Reality	The Pattern
Transformative gains in maintenance, quality, and throughput. A connected, intelligent factory floor within reach.	Pilots stall. Models drift. "AI-ready" never quite arrives. The gap between ambition and delivery quietly widens.	Transformation rarely dies in the boardroom or the proof of concept — it dies *later*, slowly, without a single dramatic failure.

Where Transformation Actually Dies

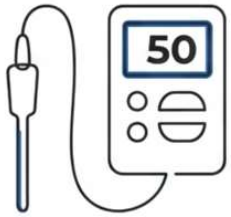
- It dies at the **seam between teams** — where everyone owns part of the problem and has veto power over the rest.
- Not a people problem - it is an **architecture and accountability problem**.
- No single failure event. Just steady, quiet erosion of trust in the data — until no one relies on it any more.

Three Barriers to Enterprise-Wide AI

Before the fix, you need the map. Every stalled transformation traces back to one or more of the same three structural failures.

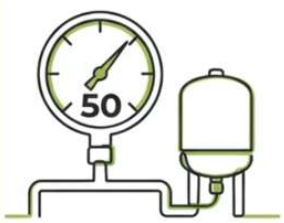
1	2	3
Inconsistent Data Structure No standardized, semantically consistent data model across the enterprise	**Governance Vacuum** No clear ownership or accountability for maintaining that model	**The Unowned Seam** The gap between OT, IT, and Security that nobody has authority to resolve

Barrier 1: Inconsistent Data Structure



50 ° C

50 Hz.



50 PSI

"50“ 50 °C, 50 Hz, or 50 PSI**?**

- Every Consuming Application Guesses & Fails Silently
- Decades Of Implicit Tag Naming, Inconsistent Units, And Undocumented Context
- Implicit Definitions Don't Scale, Can't Be Extended, And Cause Silent Misinterpretation

Barrier 2: The Governance Vacuum

Standardisation fails without an owner.

In most organisations, **no one is clearly responsible**.

- Some teams ignore it; some do it informally; a few do it well — but only inside their own silo
- Different models mean different OEE formulas. OEE something different in every plant
- Informal ownership is the default — and it always breaks under scale or personnel change

Barrier 3: The OT / IT / Security Seam

Three legitimate priorities. No owner of the space between them. This is where momentum dies.

Operations

Protects uptime above all else — any change to the control layer is a risk

Security

Protects the perimeter — every new connection is an attack surface

IT

Needs the data for analytics — and needs it now, at enterprise scale

⚠ All three are right. But no one has authority to resolve the trade-offs between them — so nothing moves.

Why AI Makes This Urgent Now

AI does not *create* insight — it **consumes structured information** and returns analysis. The quality of the output is entirely dependent on the quality of the foundation beneath it.

- Garbage structure in, confident-sounding nonsense out — drift, hallucination, silent failure
- Moving from "collecting data" to "operationalising AI" exposes *every* weakness in the data foundation

Part 2

The Diagnosis

Root Cause: The Missing Data Model

DAVID SCHULTZ – AMARACH STACKWORKS

Principal Consultant & Solutions Architect

30 years of hands-on experience spanning every layer of the automation stack — from field instrumentation at ISA-95 Level 1 through SCADA, historian and MES applications.

Level 1 – Device Layer
Instrumentation, control valves, pumps and process equipment

Level 2 – Control & Management
SCADA systems and historian applications supporting plant operations

Level 3 – MES & Enterprise
Standards-based manufacturing operations management on a modern technology stack

Industry Voice

Dozens of published LinkedIn Articles on industrial automation and MES

Standards Practitioner

ISA-95, ISA-88, ISA-101, ISA-99/IEC-62443, PackML, BPMN

The Real Foundation Isn't the AI Model

The Central Claim

The foundation of Smart Manufacturing in 2026 is the **standardised data model** — not the algorithm

What It Defines

A Master Data Model defines meaning, units, hierarchy, and governance so AI can consume data without guesswork

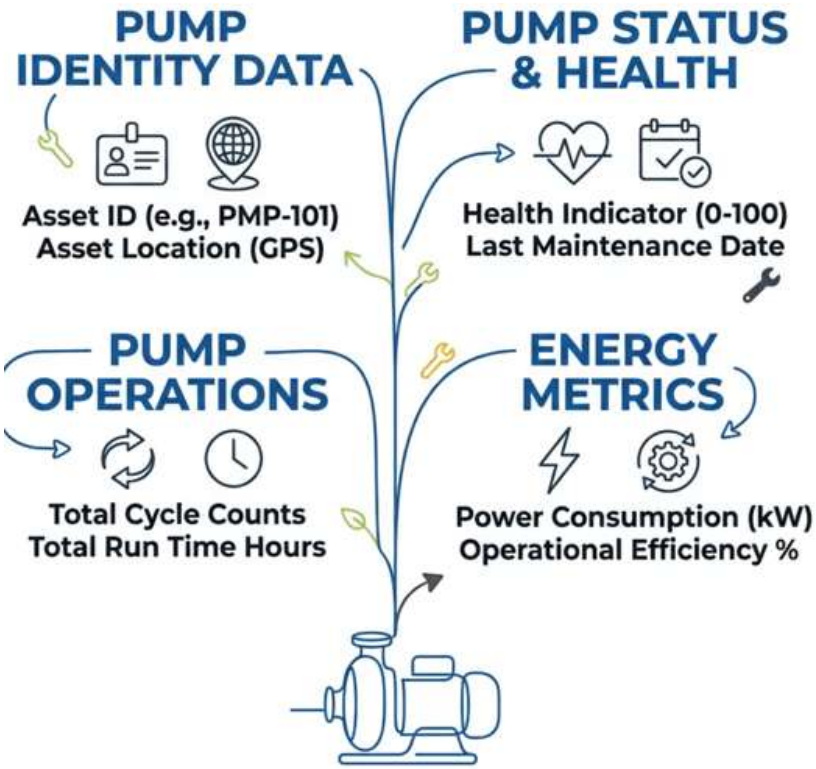
The Payoff

Fix the foundation and the analytics finally become reliable, repeatable, and scalable across every site

What Is a Manufacturing Data Model?

Not the enterprise database diagram your IT colleagues know. This lives on the **plant floor**.

A manufacturing data model is a set of related elements — variables, points, registers — in a production control system, organised to serve the people and systems that consume the data.



Example: a pump-maintenance model grouping cycle counts, run times, energy use, asset ID, and physical location — all in one coherent structure.

Simple to Complex: The Spectrum

Lower the barrier to entry — a model can start very small. The discipline is more important than the scale.

1

Single Element

A single temperature reading with defined units, range, and context is a valid, useful data model

2

Asset Model

A pump or motor grouping dozens of related elements with identity, status, and performance data

3

Hierarchical Model

Hundreds of elements organised across lines, cells, and sites — expressing sophisticated operational relationships

Hierarchy is how you express sophisticated relationships among elements. Start small; design to scale.

The Vocabulary That Prevents Confusion

Shared definitions are the prerequisite for a shared architecture. These terms will be used precisely for the rest of this talk.

Schema

The formal structure — the definition against which a model is validated

Data Model

The set of related elements — what you design, name, and govern

Instantiation

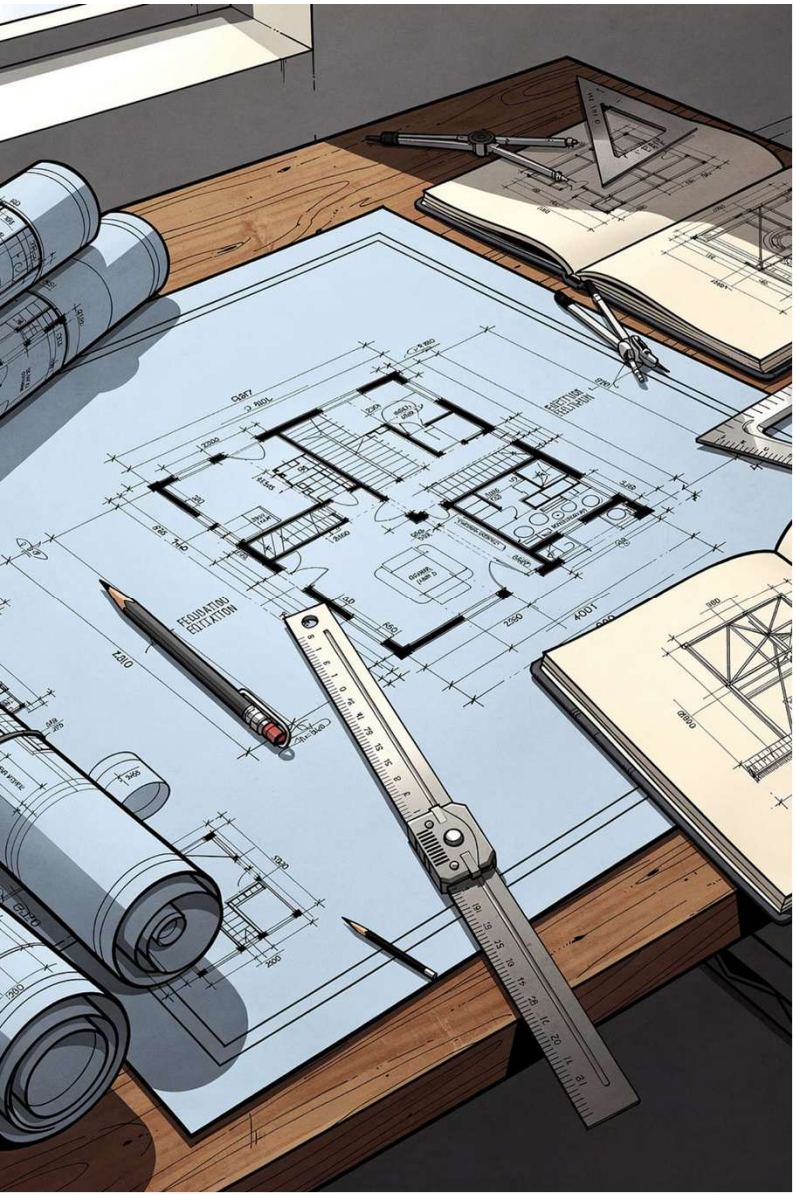
The model mapped to real tags, registers, and live values in a specific system

Data Product

An instantiated model carrying live values — ready for consumption by AI or analytics

Metadata

The descriptive context attached to each element — what makes raw data interpretable



Part 3

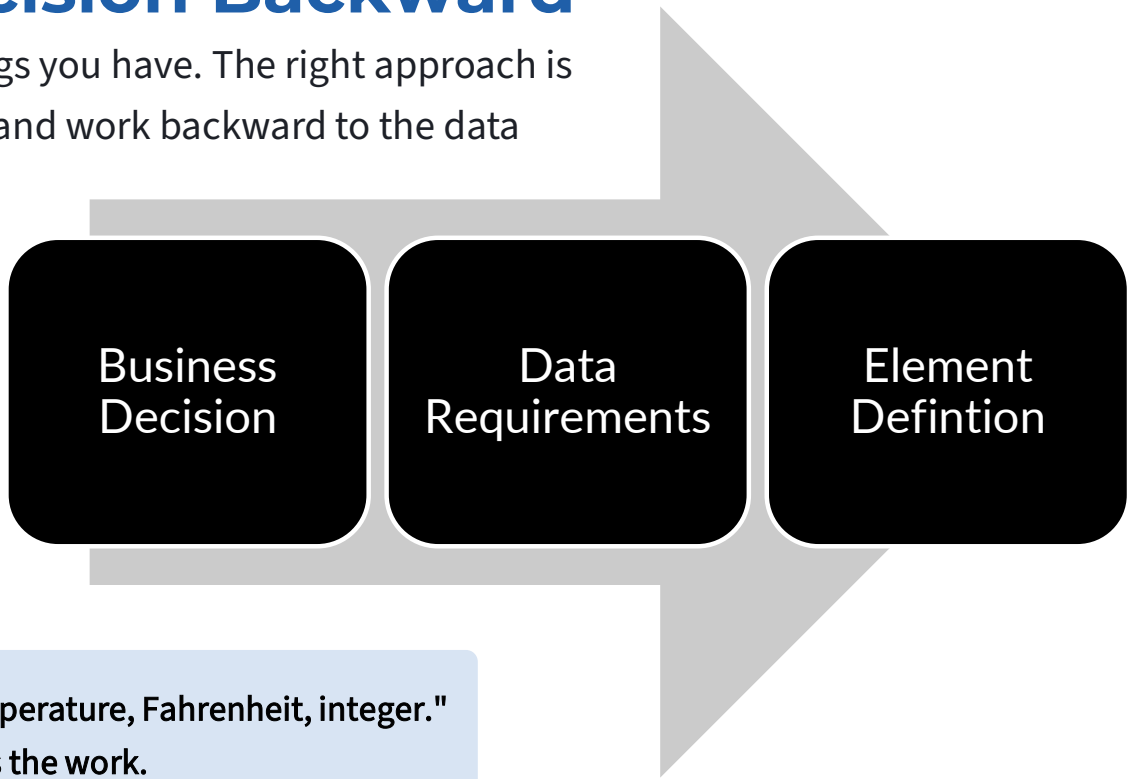
The Blueprint

Architecture, Metadata, Storage & the UNS

Architect from the Decision Backward

The most common mistake: starting with the tags you have. The right approach is to start with the **decisions that must be made** and work backward to the data required to support them.

- Define what operations and management must decide — then identify exactly what data they need
- Pin down names, units, data types, and semantics for every element before implementation



"Temp21" becomes "Oven 21 Current Temperature, Fahrenheit, integer." That transformation is the work.

Metadata: Turning Data into Information

"Register 40202" or "Tag Speed44" means nothing without context. Metadata is what makes raw tags usable — especially by AI, which cannot infer what humans have learned to assume.

Descriptive

Human-readable context: "Section 2 Discharge Pressure" — what the element represents in the real world

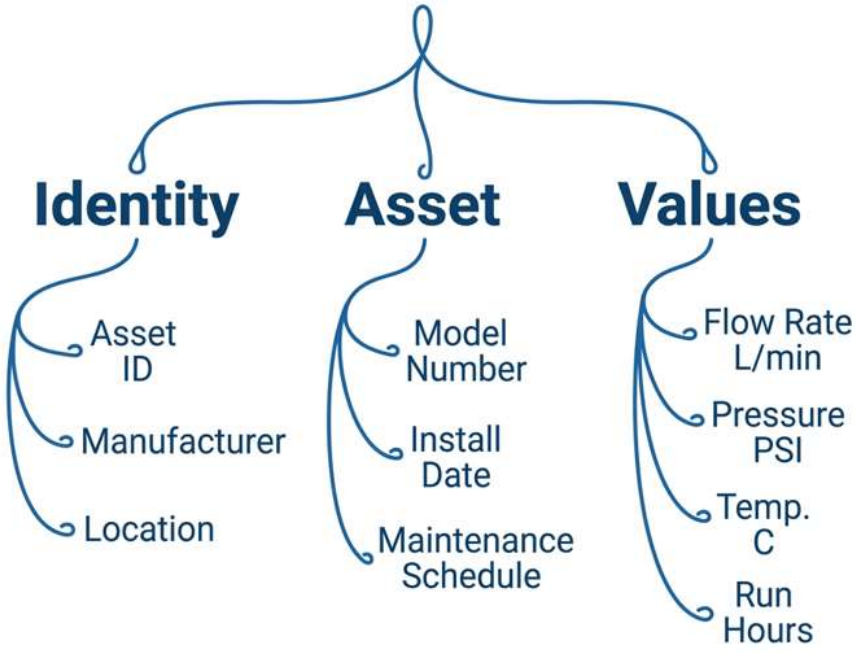
Structural

Technical definition: data type, engineering units, max/min range, array dimensions

Administrative

Operational context: last-read time, owning facility, enabled/disabled state, version

Capture metadata at model **creation** and again at **instantiation** — both points add irreplaceable knowledge.



How Data Models Are Organised

Conceptually, a model is a hierarchical list of elements plus static metadata. JSON is a clean, human-readable way to express it — and one most IT systems already understand.

Real-world implementations include:

- PLC User-Defined Types (UDTs)
- Database schemas
- Spreadsheet column hierarchies
- JSON text files

Use whatever your team will maintain. Consistency matters more than the format.

Master Data Model vs. Embedded Models

Master Data Model (MDM)

- One authoritative location — a database, SharePoint, or spreadsheet — *separate from any application*.
- Update and version once; every consuming system sees the new version immediately.

Embedded Models

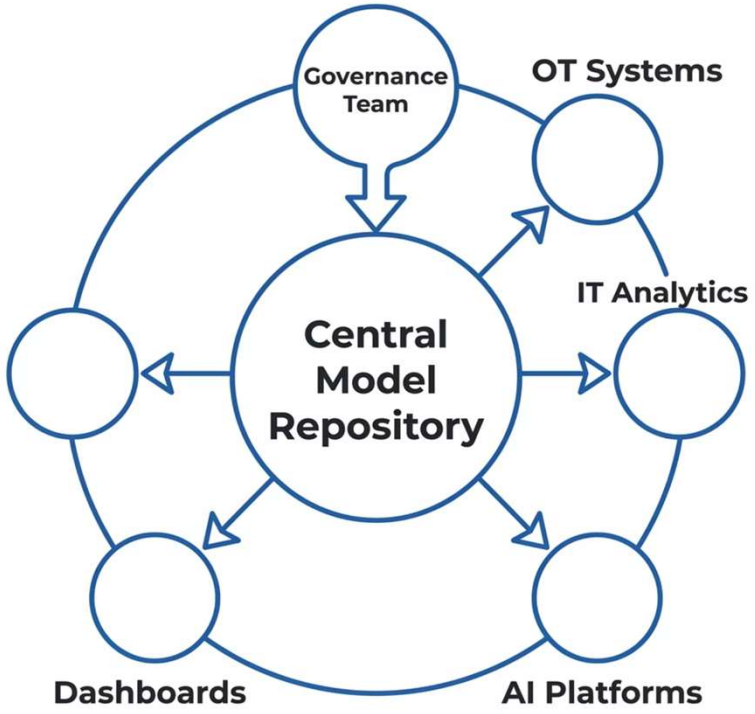
Models buried inside individual applications drift apart over time. Each system quietly develops its own interpretation — and the chaos you were trying to eliminate is recreated at scale.

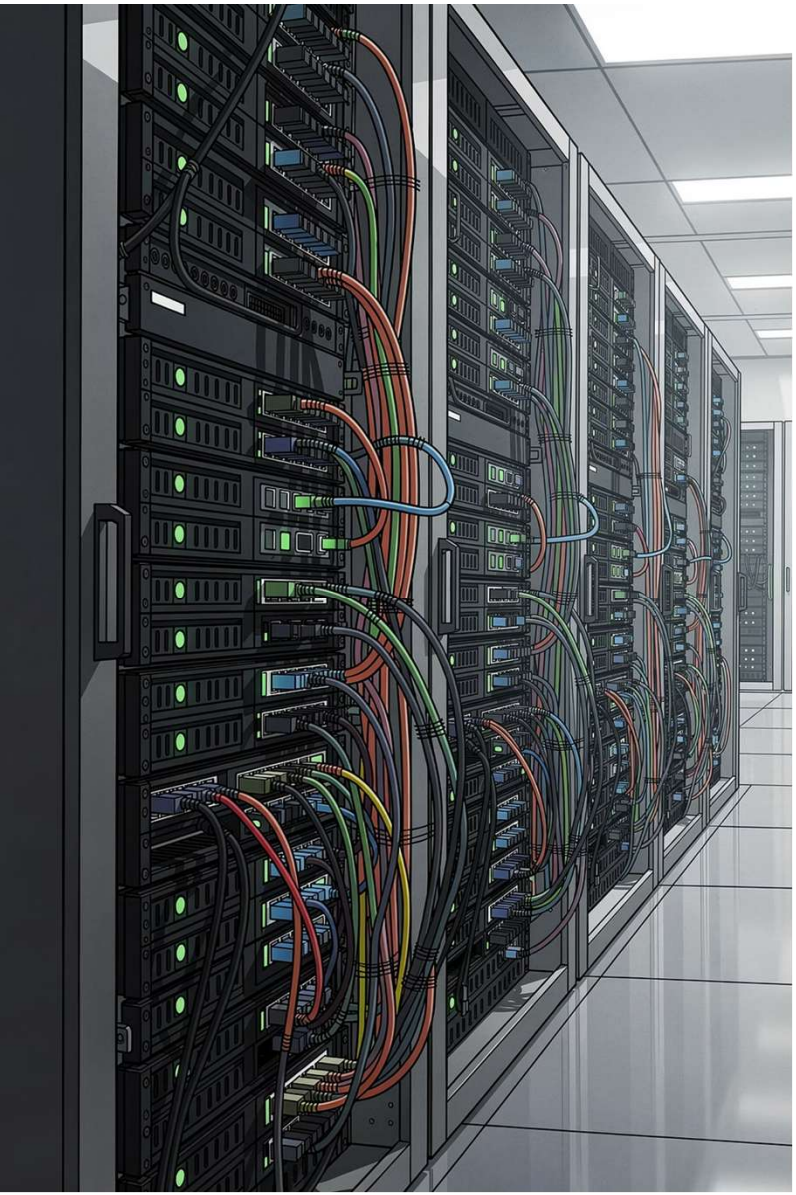
⚠ Models embedded in applications recreate the exact fragmentation you are trying to eliminate. Centralise from day one.

Where to Store Models, and How to Secure Access

Centralisation enables maintenance, versioning, and consistent consumption. Every application that needs the model must be able to reach it.

- **Write-access** strictly regulated — only authorised owners can modify or version a model
- **Read-access** made widely available — friction here breeds workarounds and shadow models
- Provide standard, IT-friendly access mechanisms (APIs, REST endpoints) regardless of storage location





The UNS Connection

A **Unified Namespace (UNS)** is a data-exchange mechanism for manufacturing data — a single logical space where all systems publish and subscribe. But a UNS without a data model is just a faster way to share ambiguous tags.

- A UNS is only effective if it exchanges **instantiated data models**, not loose tags with no semantic context
- The data model is the **payload** that makes a UNS meaningful and AI-consumable at enterprise scale

Model Both Transactional and Time-Series Data

L3 / L4 — Transactional

Traditional systems model transactional interfaces at the enterprise layer. This is well established and relatively well understood.

L1 / L2 — Time-Series

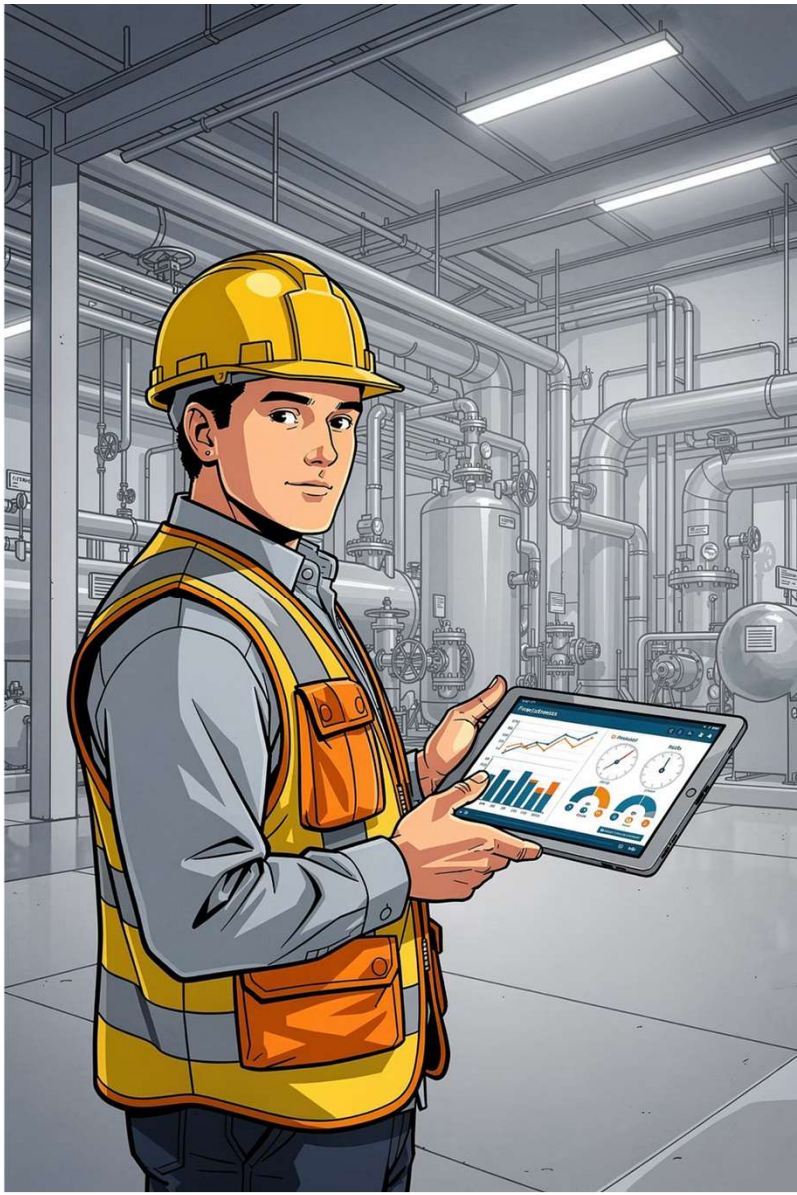
Real value is unlocked by exposing models for time-series data at the control layer. Include all PLC UDTs so AI can recognise the time-series they generate and consume them with full context.

Extending modelling discipline down to L1/L2 is where most organisations find their largest untapped AI opportunity.

Part 4

The 10 Strategies

The Actionable Core



The 10 Strategies: Overview

Ten concrete moves to turn chaotic plant data into AI-ready information that scales — grouped into five disciplines.

1. Standardise – Strategies 1-3
2. Govern – Strategy 4
3. Secure – Strategy 5
4. Evolve – Strategy 6-7
5. Expose Strategies 8-10

This is the "what do I do on Monday" list. Each strategy is actionable independently — start where the pain is greatest.

Strategies 1–3: Standardise Around the Business

1 Create company-wide standard models

- ✓ Appropriate to your operation
- ✓ Widely available — every team, every site, one source of truth

2 Use standard protocols and open APIs

- ✓ ISA-95, ISA-88, ISA-101, PackML, OpenAPI
- ✓ Use established standards and kill bespoke chaos to unlock interoperability

3 Define models around operational priorities

- ✓ Start with the problems operations must solve, not with the tags you happen to have available

Strategies 4–5: Govern and Secure

Standardisation without an owner is temporary. These two strategies directly answer Barriers 2 and 3.

Strategy 4 — Govern

Named owner, change control, approval workflow, full lifecycle management. Consistency requires accountability.

Strategy 5 — Secure

Tightly regulate write-access. Keep read-access open. Lock the authoring; liberate the consumption.

Strategies 6–7: Evolve and Extend

Models are living artefacts, not frozen documents. Plan for change from the outset.

Strategy 6 — Treat Models as Living

Add identity, description, and versioning metadata from the start. Expect elements to be added and retired as operations change — design for it, don't fight it.

Strategy 7 — Cover Both Interface Types

Model both transactional and time-series interfaces explicitly. Consider separate master models for each — the consumers and cadences are fundamentally different.

➡ Versioning is what lets you change safely without breaking downstream consumers. It is not overhead — it is insurance.

Strategies 8–9: Expose and Enrich

These are the details AI quietly depends on — and that humans routinely skip because they seem obvious.

Strategy 8 — Standard Access

Document REST APIs with Swagger/OpenAPI. For event-driven architectures, use AsyncAPI. Make interfaces discoverable and self-describing — not tribal knowledge.

Strategy 9 — Complete Metadata

Create metadata at model creation *and* at instantiation. Include deadband, enabled state, time/value quality, and unit of measure — every field an AI model needs to avoid silent error.

Strategy 10: Enforce Semantic Consistency

The capstone strategy — and the one most frequently deferred until after the damage is done.

The Rule

Enforce enterprise-wide **naming conventions, engineering units, time standards, and semantics** across every model, every site, every system

The Failure It Prevents

"Oven21Temp" vs. "Oven_21_Temperature" silently breaks AI models. Inconsistency forces custom per-site mapping — scalability dies and plant-to-plant comparison becomes impossible

Part 5

Closing the Seam

Governance as the Answer to the Slow Death

Back to the Seam: Who Owns the Model?

The data model is the **shared contract** that **OT, IT,** and **SECURITY** can all align around. It is the answer to the unowned seam — but only if you put a name on it.

- ✓ Assign explicit authority to own the model and resolve trade-offs
- ✓ **Not a committee!**
- ✓ MDM logical definitions stay authoritative;
- ✓ PLC UDTs conform to them — not the other way around
- ✓ Governance is not bureaucracy. It is the mechanism that converts a one-time pilot into a permanent capability

Implementation Checklist

A concrete, do-this-now sequence that you can take back to your organisation today.

1. Start with the business decisions — define standardised models to support those decisions
2. Attach complete metadata at model creation and at instantiation — do not defer this step
3. Store models centrally with version control — separate from consuming applications
4. Assign a named owner with explicit authority over the model and its governance process
5. Expose through secure, IT-friendly interfaces — document APIs with Swagger/OpenAPI or AsyncAPI
6. Integrate models with their instantiated data products into the UNS as the exchange mechanism

Key Takeaways

The Model Is the Foundation

AI doesn't create insight — disciplined data modelling does. The model is the foundation; the algorithm is downstream

Governance Makes It Stick

Standardisation fails without ownership. Transformation fails at the unowned seam between OT, IT, and Security

Model Intentionally, Scale Confidently

Manufacturers who model intentionally build systems that are maintainable, extensible, and genuinely AI-ready

Questions, Resources & Contact

Video Resource

"Master Data Modeling for Level 2"

Video guide to implementing L1/L2 models in practice

Full Guide

"A Solution Architect's Guide to Driving Manufacturing Insights from AI"

The complete source document behind this talk

John Rinaldi
CEO & Chief Strategist,
Real Time Automation

jrinaldi@rtautomation.com

rtautomation.com/contact-us

linkedin.com/in/johnsrinaldi

David Schultz
Principal Architect,
Amarach Stackworks

dschultz@g5ces.com

Amarach.io

linkedin.com/in/thedavidschultz

Bring your hardest data-governance and OT/IT/Security questions us The authors are available for consultations.

Appendix: FAQ — Schema vs. Data Model

BACKUP / Q&A MATERIAL

Schema

The formal structural definition — the template against which a data model is validated for correctness. It defines the rules; the model is the instantiation of those rules.

Data Model

The set of related elements designed for a specific purpose. Consumers match the model name and version to the schema to validate that the format they receive is what they expect.

Why the Distinction Matters

Conflating schema and model leads to brittle integrations. When the schema evolves, versioning ensures consumers know exactly what they are receiving — and can validate it automatically rather than discovering mismatches in production.

> A schema is a contract. A data model is an implementation of that contract. Both must be managed and versioned.

Appendix: FAQ — AI Accuracy and the Data Foundation

BACKUP / Q&A MATERIAL

AI models do not discover meaning — they depend on it being supplied. Without a disciplined data foundation, AI introduces error and drift that is difficult to detect because the outputs still *look* plausible.

Defined Units & Known Ranges

AI must know whether "50" is Celsius, PSI, or Hz — and what constitutes a normal vs. anomalous value

Quality Indicators

Time/value quality flags tell the AI whether to trust a reading or treat it as suspect — without them, stale or failed sensor data poisons the model

Asset Context & Version Traceability

Asset identity and model version traceability allow AI outputs to be audited and root-caused when behaviour drifts over time