

WHITE PAPER



The Semantic Brain

How IRIS Foundry's Cortex Knowledge Graph turns disconnected industrial data into shared context

Contents

The Context Gap	3
Drowning in Data, Starving for Context	5
What Contextualization Actually Means	8
Cortex: The Industrial Entity Graph	14
The Neural Graph: A Structure That Learns	22
Building and Maintaining the Graph	25
The Industrial LLM: Language Grounded in the Graph	27
From Context to Action: Agentic AI	29
Governance, Security and Operational Trust	31
Use Cases in Practice	33
Deployment and Adoption	35
Measuring Value	37
In Summary	39
Appendix A — Glossary	41
Appendix B — Sources and Notes	43

01

The Context Gap

A refinery unit trips at 03:40. The board operator sees a pressure alarm on a compressor discharge. Within ninety seconds they need to know several things at once: whether this pattern has occurred before, what the vendor limit for this machine actually is, whether the last inspection flagged anything, which downstream units are about to be affected, and what the standing procedure says to do next.

Every one of those answers exists somewhere in the organization. The historical pattern is in the historian. The vendor limit is in a PDF in an engineering repository. The inspection note is in the maintenance system. The downstream dependency is in a P&ID. The procedure is in a document management system, possibly in a revision the operator has not seen.

None of those systems knows about the others. The operator becomes the integration layer — assembling context from memory, experience and whatever they can retrieve in the time available. When the operator is experienced, this works remarkably well. It is the single most valuable and least documented capability in the plant.

The most sophisticated contextualization engine in most plants is a person with twenty years of service and no succession plan.

1.1 Why this is now a strategic problem

Three pressures have converged to make human contextualization untenable as an operating model.

01 The knowledge is leaving

Deloitte and The Manufacturing Institute project that 2.4 million manufacturing roles may go unfilled by 2028, with more than 2.6 million baby boomers retiring from manufacturing positions over the decade. Eighty-nine percent of manufacturing executives confirm a talent shortage. The exposure is not headcount alone — it is that the departing cohort holds the undocumented relational knowledge that makes the data usable. When a thirty-year reliability engineer retires, the plant does not lose a pair of hands. It loses the index.

02 The cost of a slow answer has risen

Fluke's 2025 study of more than 600 senior manufacturing decision-makers across the United States, United Kingdom and Germany found unplanned downtime costing the sector up to \$852 million weekly, at an average of \$1.7 million per hour. Sixty-one percent of manufacturers experienced unplanned downtime in the preceding year; 48% report six to ten incidents weekly. In that economics, the twenty minutes an engineer spends assembling context is not an administrative overhead. It is the dominant term in the cost of the event.

03 AI has raised the stakes on data quality

Generative AI made it obvious that ungrounded systems produce fluent, confident, wrong answers. An LLM asked about a specific compressor will answer — plausibly, and without any means of knowing whether the answer applies to this machine, this configuration or this revision of the procedure. The industrial requirement is not fluency. It is traceability: an answer that can be followed back to the tag, the document and the work order it came from. That requirement is a data architecture requirement before it is a model requirement.

~90% of industrial data never used

\$1.7M per hour of unplanned downtime

2.4M roles unfilled by 2028

Figure 1-1. The three pressures that make contextualization urgent: unused data, expensive downtime, and the loss of experiential knowledge as the workforce turns over.

This is not an argument that historians, data lakes, MES or ERP systems were mistakes. Each solved the problem it was built for, and each remains the system of record for its domain. The argument is narrower and more specific: none of them was designed to hold the relationships between domains, and no amount of aggregation produces those relationships automatically. A data lake with everything in it and no relationships is a larger version of the same problem.

Nor is this an argument for replacement. Cortex — the entity graph described in this paper — reads from source systems and writes back to them where appropriate. It does not ask an organisation to migrate its historian or re-platform its ERP. Contextualization is a layer, not a rip-and-replace.

02

Drowning in Data, Starving for Context

A single large industrial site typically generates hundreds of thousands of measured values across dozens of applications. That estate is heterogeneous in every dimension that matters: in structure, in vocabulary, in update frequency, in ownership, and in the assumptions each system makes about what an "asset" is.



2.1 Five kinds of fragmentation

01 Structural fragmentation

Time-series data, relational records, unstructured documents, geometric models and imagery each require a different storage engine and a different query paradigm. A question that spans them — "show me the vibration trend for every pump that had a seal replacement in the last year and appears on this P&ID" — cannot be expressed in any single system's query language.

02 Semantic fragmentation

The same physical pump is P-204 in the historian, 23-P-0204-A in SAP, "Charge Pump A" in the operating procedure, and an unlabelled symbol on a drawing. No system is wrong. There is simply no authority that declares these four references to be the same object.

03 Temporal fragmentation

Historian data updates in seconds. Work orders update in days. Drawings update when someone remembers, sometimes years after the modification they document. A "current" view assembled from all three is current in only one of them.

04 Organisational fragmentation

Reliability owns the condition data. Operations owns the process data. Maintenance owns the work order data. Engineering owns the drawings. Each has legitimate access controls, and each control was designed without reference to a cross-domain question.

05 Interpretive fragmentation

A process variable cannot be interpreted in isolation. Is it trending as expected for this operating mode? What is the vendor-recommended maximum? Was the equipment recently inspected, and what was observed? The number is meaningless without the surrounding facts — and the surrounding facts live in four other systems.

THE COMPOUNDING EFFECT

Each type of fragmentation is individually tractable. An integration project can reconcile two systems. A naming convention can address semantic drift within one plant.

What defeats point solutions is that the five compound. Reconciling names does not fix temporal skew. Fixing temporal skew does not grant cross-domain access. And every fix is a snapshot that begins decaying the moment the plant changes — which, in a working facility, is continuously.

2.2 Where the effort actually goes

The visible symptom is time allocation. Subject matter experts describe spending the majority of an analysis cycle finding, cleaning and reconciling data, and a minority actually interpreting it. The pattern is consistent across industries and independent of how much has been invested in data infrastructure, because the investment typically improved availability rather than meaning.

The sequence below is a composite of workflows observed repeatedly in reliability and process engineering across refining, chemicals, mining and discrete manufacturing.

HUMAN CONTEXTUALIZATION – a typical root-cause cycle

- | | |
|---------------------------------|---------------------------------------|
| 1. Form hypothesis | → (engineer, from experience) |
| 2. Locate the P&ID | → document repository, ~20 min |
| 3. Identify the correct tag | → historian, naming mismatch, ~15 min |
| 4. Pull the trend | → historian export to spreadsheet |
| 5. Retrieve equipment history | → maintenance system, separate login |
| 6. Find technical documentation | → vendor PDF, possibly superseded |
| 7. Reconcile the identifiers | → manual, error-prone, undocumented |
| 8. Assemble and interpret | → spreadsheet, one-off, not reusable |

Elapsed: hours to days. Reusable next time: none.
Institutional memory retained: only in the engineer.

Figure 2-1. The manual contextualization cycle. Every step is a system boundary crossed by a human, and the resulting assembly is discarded once the question is answered.

Two properties of this cycle deserve emphasis. First, the output is not reusable — the next engineer investigating a related failure repeats the entire sequence. Second, the reconciliation performed at step seven is the single most valuable artefact produced, and it is never captured. The organisation pays for that reconciliation every time and keeps it no time.

2.3 Why aggregation is not contextualization

The common institutional response is to centralise: build a lake, land everything in it, and expect that proximity will produce insight. Proximity is genuinely useful — it removes the access problem and the export-to-spreadsheet step. But a lake stores objects, not relationships. After the migration, the pump is still four different identifiers; the drawing is still an unindexed PDF; the temporal skew is unchanged. The engineer still performs step seven.

Unified namespaces improve on this materially by imposing a consistent hierarchy and consistent naming, and they are a genuine prerequisite. But a namespace expresses containment — this sensor belongs to this asset, which sits in this unit.

It does not express the wider relational fabric: what this asset feeds, what fails when it degrades, which procedure governs it, which failure modes it exhibits, which KPI it moves. Those are graph relationships, and they require a graph.

Approach	What it solves	What remains unsolved
Historian	High-fidelity time-series capture and retrieval at scale	No link to documents, work orders, failure modes or business context
Data lake / warehouse	Centralised storage; removes access and export friction	Stores objects, not relationships; identity reconciliation still manual
Unified namespace	Consistent hierarchy and naming across sources	Expresses containment only; no causal, procedural or failure-mode relationships
Point integration	Reconciles two named systems for a defined use case	Does not generalise; decays as either system changes
Knowledge graph	Persistent, queryable relationships across all domains	Requires deliberate construction and continuous maintenance

Table 2-1. Each layer is necessary. Only the graph layer holds relationships, which is what cross-domain questions actually require.

03

What Contextualization Actually Means

Contextualization is the automated discovery and maintenance of relationships between data elements that are connected in the physical world but not explicitly represented in any system that records them.

The definition carries three commitments worth separating.

3.1 Relationships, not records

The unit of value is the edge, not the node. That a pump exists is recorded in five systems already. That this pump feeds that exchanger, is governed by this procedure, exhibits these failure modes, and contributes to that throughput KPI — none of these is recorded anywhere as a machine-readable fact. Those edges are what make a question answerable without a human in the loop.

The distinction has a practical consequence: the value of a graph grows super-linearly with the relationships captured, not linearly with the data ingested. Adding a tenth data source to a lake adds a tenth of a lake. Adding a tenth data source to a graph creates edges to the nine already present.

3.2 Automated, because manual does not scale

Relationships can always be built by hand. Reliability engineers do it constantly, in their heads and in spreadsheets. The problem is arithmetic: a mid-sized plant has tens of thousands of assets and hundreds of thousands of tags, and the relationships between them number in the millions. Manual reconciliation of a single unit is a multi-week exercise; manual reconciliation of a site is a multi-year one, by which point the early work is stale.

Automated contextualization uses several techniques in combination, because no single technique covers the range of industrial data:

- **Pattern and fuzzy matching** — reconciling identifiers that differ by convention, padding, prefix or delimiter across systems.
- **Structural extraction** — parsing P&IDs and engineering drawings to recover asset symbols, tag references, connectivity and drawing metadata.
- **Master-data reconciliation** — aligning equipment master records, functional locations and BOM structures against the physical asset hierarchy.
- **Semantic embedding** — chunking and embedding procedures, manuals and reports so that text is retrievable by meaning rather than by filename.
- **Domain-model inference** — applying a library of asset and failure-mode templates so that a newly identified centrifugal pump inherits the failure modes and monitoring logic already established for its class.

The pre-built asset model library in IRIS Foundry includes predefined failure-mode patterns covering more than 85% of common industrial assets, which reduces new asset model setup from days to minutes and means a plant does not begin from an empty ontology.

Three of these techniques carry enough of the load — and are misunderstood often enough — to deserve deeper treatment: entity matching, interactive diagram creation, and imagery contextualization.

3.3 Entity matching: the keystone discipline

Entity matching is the process of establishing, with evidence, that records held in different systems refer to the same physical object — that P-204 in the historian, 23-P-0204-A in the ERP equipment master, "Charge Pump A" in the operating procedure and an unlabelled symbol on drawing ISO-14424 are one pump.

It is worth being precise about why this is difficult, because the difficulty is routinely underestimated at project scoping. Naming conventions differ not only between systems but between eras: a plant built in phases by different EPC contractors carries each contractor's convention permanently. Identifiers diverge through padding, prefixes, delimiters and transcription. Acquired sites arrive with entirely foreign schemes. Equipment masters are partially populated — the normal condition, not the exception — and some correspondences are genuinely ambiguous, resolvable only by someone who knows the plant.

IRIS Foundry approaches matching as a layered pipeline rather than a single algorithm, with each layer handling what the previous one could not:

- **Deterministic and pattern-based matching** — exact and rule-derived correspondences resolved first, cheaply and with certainty, clearing the bulk of the estate.
- **Fuzzy and structural matching** — identifiers that differ by convention, padding or delimiter, matched with confidence scores rather than binary verdicts.
- **Semantic matching** — named-entity recognition and contextual embeddings that align tags by meaning, so "discharge pressure" and "outlet press" reconcile even when no string similarity exists. Ontology alignment maps industry-specific vocabulary across systems.
- **Graph-based entity linking** — candidate matches evaluated against the relationships already in the graph: if two candidate records claim to be the same pump but sit in different process areas, the structure itself vetoes the match.
- **Anomaly detection over the mappings** — the match set itself is monitored for conflicts and outliers, so a bad mapping is flagged rather than silently propagated.

Low-confidence matches are routed to human review, and every confirmation or correction is retained as training signal — the pipeline learns the plant's conventions and improves with each pass. This closed loop is what distinguishes a matching capability from a matching project.

WHY ENTITY MATCHING MATTERS MORE THAN IT APPEARS TO

Identity is the load-bearing layer of the entire graph. Every cross-system answer — every traced root cause, every copilot response, every agent action — silently depends on the assertion that these records describe the same object. An identity error does not stay local: it propagates into every answer that touches the mis-matched asset, and it does so invisibly.

Matching is also where the economics of contextualization are decided. The reconciliation that engineers perform by hand at step seven of every manual investigation is entity matching, paid for repeatedly and captured never. Performing it once, with evidence and confidence scores, converts a recurring cost into a permanent asset — and it is why reconciliation of a first scope is measured in weeks while subsequent sites are measured in days.

3.4 From static drawings to interactive diagrams

Process and instrumentation diagrams are the densest repository of relational truth in most plants: they encode connectivity, instrumentation, isolation points and process flow — relationships that are recorded nowhere else. They are also, in most organisations, flat PDFs: visible to a human, invisible to every system.

IRIS Foundry's P&ID ingestion converts the drawing set from documentation into infrastructure. AI-assisted extraction recovers assets and sensors from uploaded drawings and infers the connections between them; assets and tags are mapped to drawing numbers, plant systems and subsystems; connected equipment is followed across drawing boundaries, so a line that exits one sheet and enters another is a single traversable path rather than two dead ends. Elements the parser does not identify are bounded and defined by the user, and every proposed connection is verifiable and correctable before it enters the graph — a deliberate human checkpoint at the exact place where an extraction error would be most expensive.

The result is not a better document viewer. Once extracted, the drawing becomes an interface to the graph: selecting a symbol retrieves the live tag values, health score, open work orders, alerts and associated documents for that equipment;

traversing the drawing upstream or downstream is traversing the plant's actual dependency structure. The engineer's native medium — the drawing they would have pulled anyway — becomes the fastest entry point into everything the graph knows.

- **Investigation speed** — root cause work follows physical connectivity. When upstream and downstream are clickable rather than reconstructed from memory, tracing a disturbance takes minutes.
- **Isolation and permit planning** — identifying isolation points and affected equipment from a live diagram removes a class of error from permit-to-work preparation.
- **Brownfield onboarding** — for older sites, drawings are frequently the only machine-recoverable record of connectivity. P&ID extraction is what makes a decades-old plant contextualizable at all.
- **Institutional memory** — the mental map senior engineers carry of "what feeds what" is externalised into a structure any colleague can navigate.

3.5 Contextualizing imagery data

Imagery is the fastest-growing category of industrial data and the least contextualized. Inspection photographs, thermal captures, drone surveys and fixed production cameras generate volumes that dwarf most tabular sources — and in most organisations they accumulate in folders, unlinked to the assets they depict.

The distinction the graph makes is simple: an image without context is a picture; an image linked to an asset, a location, a timestamp and a work order is a condition record. The first is reviewed once and forgotten. The second joins the asset's history, comparable against every earlier capture of the same equipment, retrievable by anyone investigating that asset years later.

IRIS Foundry applies computer vision across this estate and writes the results into the graph rather than into a separate silo:

- **Integrity and corrosion detection** — vision models identify corrosion, including corrosion under insulation, from inspection imagery, feeding risk-based inspection prioritisation so inspection effort concentrates where the evidence points.
- **Production quality at line speed** — vision-based detection of foreign material and defects drives configurable reject rules by class, size, confidence and location, without reducing line speed. Each detection creates traceable evidence for supplier corrective action and quality improvement.
- **Safety monitoring** — where camera feeds are provided: PPE compliance, zone access, hot-work and vehicle safety, with real-time alerts routed through the same governed channels as every other event.

Two properties of the implementation matter as much as the models. First, the loop is closed: production images, detections and operator annotations become governed training and test data for future model releases, so accuracy compounds from operations rather than depending on periodic vendor retraining. Second, the models are drift-aware — changes in lighting, optics, product appearance or supplier material trigger monitoring, review and retraining workflows, which is the difference between a vision system that works at commissioning and one that still works two years later.

THE COMMON THREAD

Entity matching, diagram extraction and imagery contextualization look like three separate features. They are one discipline applied to three media: taking information that exists only in a form humans can read — a naming convention, a drawing, a photograph — and converting it into relationships a machine can traverse.

Every relationship so converted stops being paid for repeatedly and starts compounding. That is the entire economic argument for contextualization, applied at the level of individual techniques.

3.6 Continuous, because plants change

The third commitment is the one most often missed. A plant is not a fixed object. Equipment is replaced with non-identical units. Procedures are revised. Process flows are reconfigured seasonally. Sites are acquired with entirely foreign naming conventions. A contextualization effort delivered as a project produces an artefact that is accurate on the day it ships and progressively less accurate every day after.

Continuous contextualization treats the graph as a living structure: new data is contextualized on arrival against the existing model, new relationships are proposed and either confirmed or corrected, and the model itself is enriched by use. Each correction an engineer makes is retained, so the graph converges toward the plant rather than drifting away from it. Section 5 examines this learning behaviour in depth, because it is the property that separates a knowledge graph from every integration approach that preceded it.

3.7 A worked example

Consider the compressor trip from Section 1, asked of a contextualized graph rather than a person. Figure 3-1 shows the answer as Cortex assembles it: a traversal of labeled edges radiating from the alarm and the asset it measures.

Graph traversal – compressor trip, 03:40

One alarm, one asset - every fact needed for the decision, one labeled edge away

Answer assembled in seconds - every element traceable to source

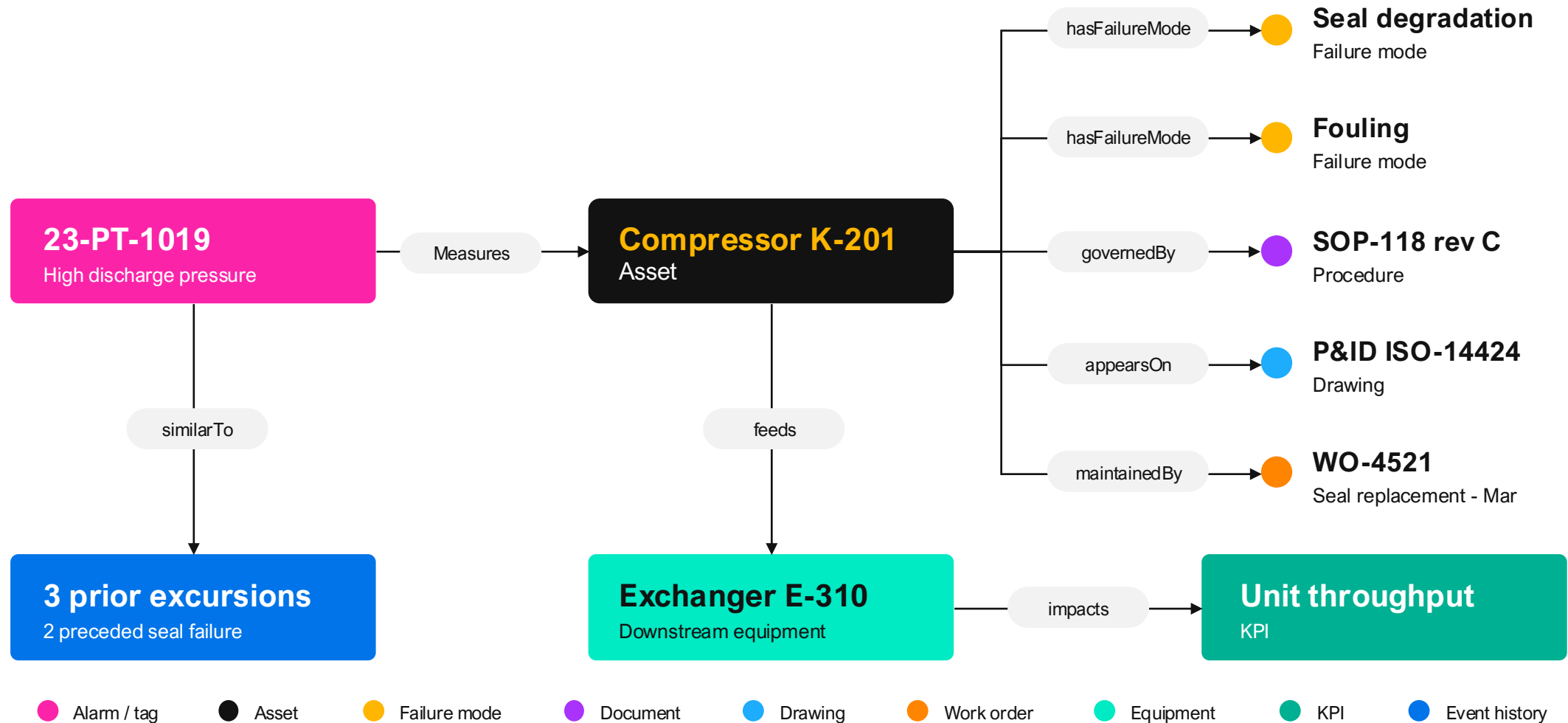


Figure 3-1. The compressor trip as an entity-graph traversal. One alarm, one asset — and the failure modes, governing procedure, drawing, maintenance history, downstream equipment and threatened KPI, each one labeled edge away.

The same traversal, written out structurally:

GRAPH TRAVERSAL – the same question, answered structurally

ALARM: high discharge pressure, tag 23-PT-1019

```

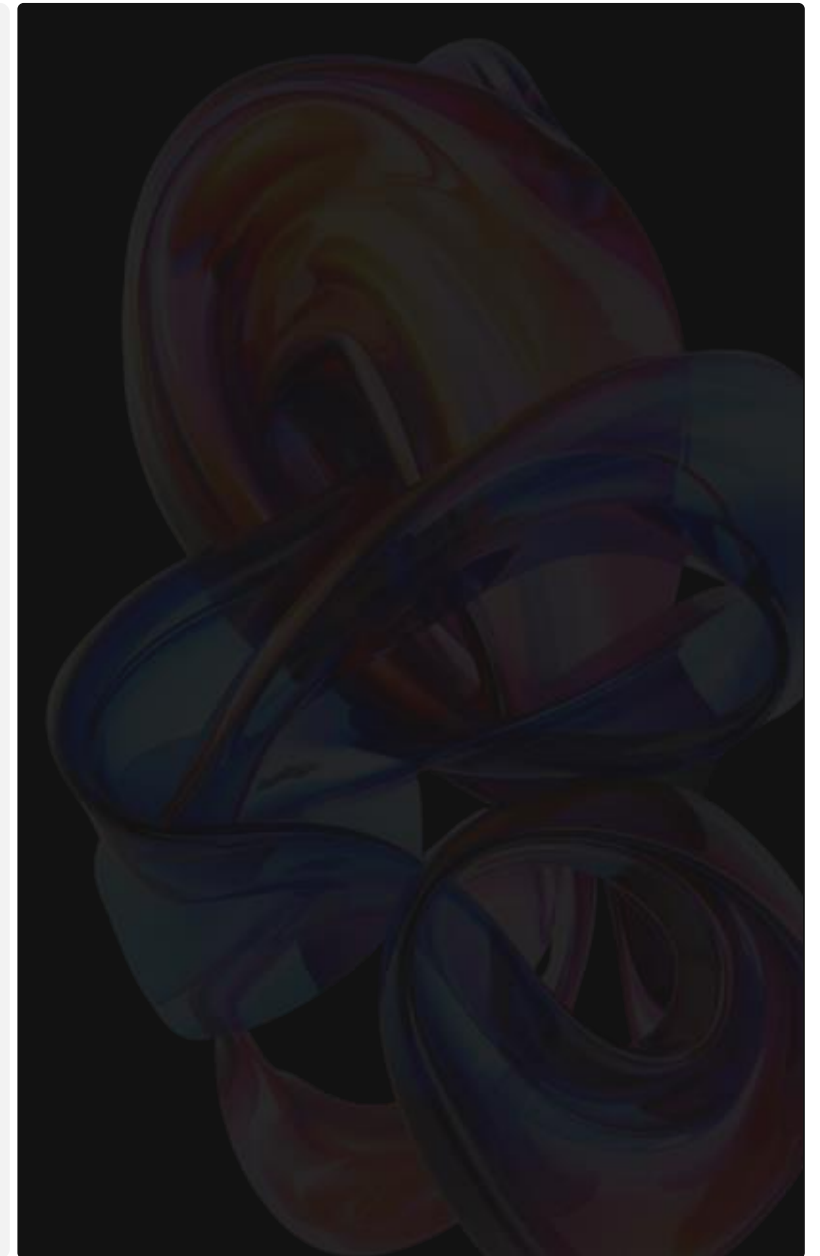
|
+-- [MEASURES] -----> Asset: Compressor K-201
|
|                               |
|                               +-- [HAS_FAILURE_MODE] -> Seal degradation
|                               |                               Fouling
|                               +-- [GOVERNED_BY] -----> SOP-118 rev C
|                               +-- [APPEARS_ON] -----> P&ID ISO-14424
|                               +-- [MAINTAINED_BY] ----> W0-4521 (seal, Mar)
|                               +-- [FEEDS] -----> Exchanger E-310
|                               |
|                               +-- [IMPACTS] -> Unit throughput
|
+-- [SIMILAR_TO] ---> 3 prior excursions, 2 preceded seal failure

```

Answer assembled: < 2 seconds. Every element traceable to source.

Figure 3-2. The traversal in structural form. The relationships were established once and are reused by every subsequent question that touches this equipment.

The graph does not make the engineer redundant. It removes the seven steps of retrieval and reconciliation, and returns them to the one step that requires judgement — deciding what to do. It also produces an answer that is inspectable: every element carries a link to the system it came from, which is the property that makes the output admissible in a safety-critical review.



04

Cortex: The Industrial Entity Graph

Cortex is IRIS Foundry's knowledge graph — an open, flexible, labeled entity graph that represents an operation as it actually is. Every asset, machine, sensor, process, event, work order, BOM line, defect, inspection, customer order, document, drawing, operator and KPI is a node; every relationship between them is a labeled, traversable edge.

Each word in that description carries weight. Labeled means edges have meaning, not just existence — FEEDS, MEASURES, GOVERNED_BY, IMPACTS — so a traversal is a chain of reasoning, not a chain of joins. Open means the graph is reachable through documented APIs by any consumer the organisation authorises, rather than being the private substrate of one vendor's applications. Flexible means the model is not a fixed ontology imposed on the plant: organisations model their own industrial reality, and the same entities can participate in as many hierarchies and perspectives as the operation needs — locational, functional, process-flow, organisational — without duplication.

That last property matters because there is no single representation that serves every consumer. A maintenance planner, a process engineer and an energy analyst need different shapes of the same plant. A graph that forces one canonical hierarchy fails two of the three; an entity graph lets each perspective be a projection, while the underlying entities and evidence stay singular.

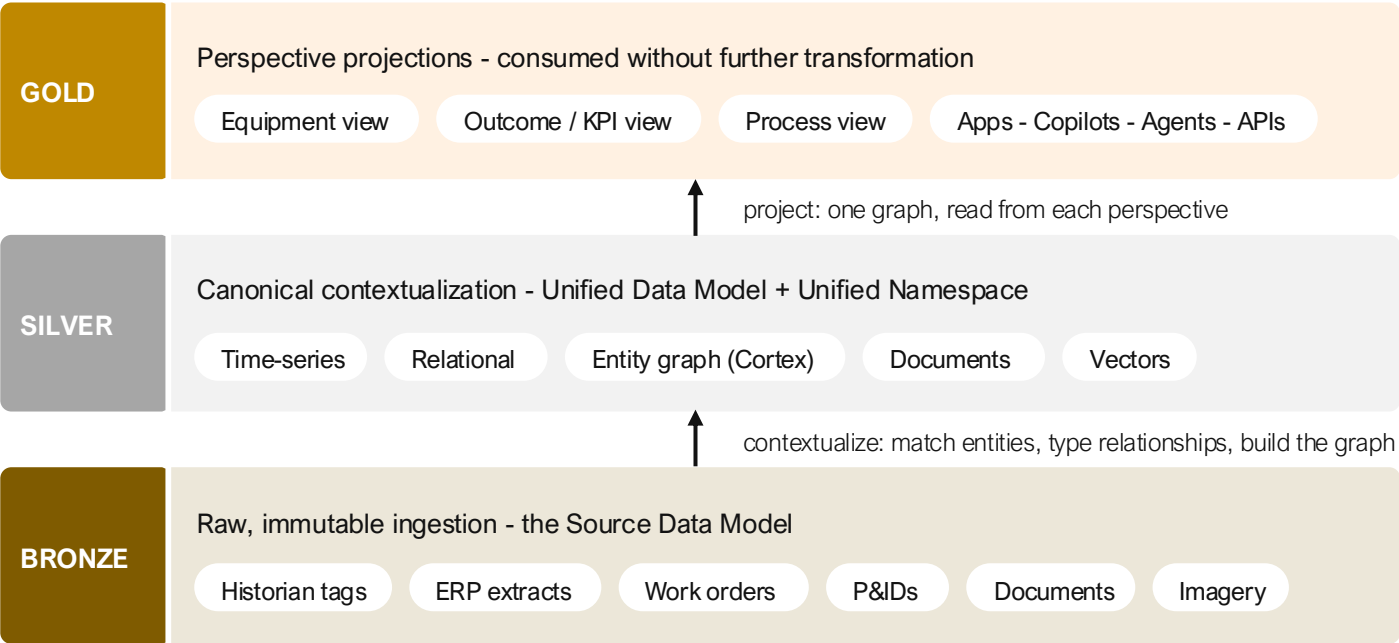
In the vocabulary of modern data architecture, what Cortex provides is an operational ontology: a shared, machine-readable model of the organisation's entities, the relationships between them, and — as Section 4.6 develops — the actions that can be taken on them. The term matters less than the discipline it names: agreeing once, explicitly, on what things are and how they connect, so that every application, model and agent afterwards reasons over the same world instead of each reconstructing its own.

4.1 The medallion progression

IRIS Foundry organises data in three progressive layers. The separation matters because it keeps raw evidence auditable while allowing the semantic model above it to evolve without reprocessing history.

From evidence to decisions - the medallion progression

Each layer earns the one above it; nothing is discarded on the way up



Lineage runs the other way: every assertion above traces to an untransformed original below.

Layer	Contents	Purpose
Bronze	Raw, immutable ingestion from every source — historian tags, ERP extracts, documents, imagery, engineering files	Evidence and lineage. Nothing is discarded; every downstream assertion can be traced to an untransformed original.
Silver	Cleansed and canonical data stored in the engine best suited to its type: TimescaleDB for time-series, PostgreSQL for relational, graph store for relationships, document store for unstructured, vector store for embeddings	The Unified Namespace contextualizes all of it against a common asset hierarchy with associated equipment, process and business information.
Gold	Domain-optimised products: materialised views, domain aggregates, and the perspective-specific projections of the Cortex entity graph	Direct consumption by analytics, applications, copilots and agents without further transformation.

Table 4-1. The Bronze–Silver–Gold progression from raw immutable ingestion through canonical contextualization to application-ready data models.

A design consequence worth noting: because the graph lives in Silver and Gold rather than in the source systems, no source system is modified to participate. A historian that has run untouched for fifteen years continues to run untouched.

4.2 The Unified Namespace

The Unified Namespace is the canonical addressing layer — the mechanism by which a tag, a work order, a document and a drawing symbol are established as references to the same physical thing. IRIS Foundry aligns the namespace to recognised industrial standards, including ISA-95 and ISA-88 for manufacturing hierarchy and, in energy deployments, OSDU.

Standards alignment is not ceremony. It determines whether a model built at one site transfers to the next. When the namespace is standards-aligned, an anomaly detection model trained on a crusher at one mine can be applied to a comparable crusher at another with reconfiguration rather than rebuilding — which is what converts a successful pilot into a fleet programme.

Data type	Storage engine	Contextualization mechanism
Time-series (historian, DCS tags)	TimescaleDB	UNS maps tags to the physical equipment hierarchy
Relational / transactional (work orders, POs)	PostgreSQL	Master-data reconciliation against the asset hierarchy
Documents (SOPs, manuals, reports)	Document / NoSQL store	Metadata tagging with permission-aware indexing
Semantic relationships	Graph database	Equipment, process and business context linked as a graph
Retrieval embeddings	Vector store	Text chunks embedded for semantic search and retrieval

Table 4-2. Polyglot persistence in the Silver layer. Each data type is stored in the engine suited to it, and the Unified Namespace is what makes them jointly addressable.

4.3 Source Data Model and Unified Data Model

Within the medallion progression, IRIS Foundry maintains two deliberately distinct representations of the estate, and the discipline of keeping them separate is one of the platform's most consequential architectural decisions.

The Source Data Model

The Source Data Model is the data as its system of record shaped it — historian tags with their native naming, ERP equipment masters with their functional-location structure, documents with their repository metadata — liberated from

those systems and made available in its original state, ingested immutably at the Bronze layer.

Preserving the source shape is not conservatism; it earns its keep four ways. It provides lineage — every downstream assertion traces to an untransformed original, which is what makes a graph answer admissible in an incident review. It respects authority — the source system remains the system of record, and the platform never pretends otherwise. It prevents information loss — transformation applied at ingestion destroys whatever the transform did not anticipate needing, and no one anticipates everything. And it enables re-derivation — when the unified model evolves, history is reprocessed from originals rather than being stranded in an obsolete schema.

The Unified Data Model

The Unified Data Model is the canonical representation built above it at the Silver layer: a single, standards-aligned set of schemas — ISA-95 and ISA-88 for manufacturing hierarchy, OSDU in energy deployments — that defines how assets, sensors, failure modes, work orders, documents and production context relate, regardless of which system each fact arrived from. Every data type flowing in from historian, CMMS, ERP and laboratory systems is mapped onto this one consistent structure, and the Unified Namespace gives everything in it a single address.

The UDM is where the plant stops being a collection of exports and becomes a model. It is also a template: once established for a first scope, subsequent units and sites inherit it, which is the mechanism behind the weeks-to-days scale-up economics described in Section 6.

The UDM is not an abstraction confined to architecture diagrams. Every entity in Cortex carries a UDM type, and the type system is the working surface of the product: the graph explorer lists the operation's UDM types with live instance counts — assets, machines, machine cycles, BOMs and BOM lines, calibration records, customers, customer orders and specifications, defects, design documents, inspections, labor records, health alerts, purchase orders, shipments — and each type is a lens that can be switched on or off. The schema an architect designs is, unchanged, the vocabulary an engineer browses.

Concern	Source Data Model	Unified Data Model
Shape	As the source system holds it, unmodified	Canonical, standards-aligned (ISA-95, ISA-88, OSDU)
Purpose	Evidence, lineage, re-derivation	Querying, contextualization, application consumption
Authority	The source system remains the system of record	The platform is the system of intelligence
Change behaviour	Append-only; history is never rewritten	Evolves as the model improves; rebuilt from source when needed
Who consumes it	Auditors, lineage tooling, reprocessing pipelines	Applications, copilots, agents, analytics — everything built on Cortex

Table 4-3. Two representations, two jobs. Collapsing them into one — the default in most integration projects — sacrifices either auditability or usability.

Why the separation is the architecture

Applications, copilots and agents consume the Unified Data Model exclusively, never the source representations. That decoupling is what makes the estate changeable: a historian migration or an ERP upgrade becomes a revision to one set of mappings, not a change to every application built on the platform. It is also what makes cross-site analytics honest — a model trained against the UDM at one site applies at another because both sites present the same schema, whatever their underlying systems.

Most integration projects collapse these two models into one — transforming at ingestion and keeping only the transformed result. The consequence surfaces slowly: the transform embeds today's assumptions, the original is gone, and three years later the organisation is re-extracting from source systems to answer a question the schema did not anticipate. Keeping both models is the cheaper decision on every horizon longer than the project that made it.

4.4 One graph, every perspective

Because Cortex holds entities rather than pre-committed views, the perspectives an operation needs are traversals over the same nodes and edges — never separate models that drift apart and demand reconciliation.

The equipment perspective

Starting from any asset node, everything known about that equipment is one hop away: health scores, predictive recommendations, alerts, live time-series, maintenance history, open work orders, documents, P&IDs and inspection imagery — retrieved together, related to each other, and traceable to source. A process-flow traversal over the same entities gives the flow-of-operations view: what feeds this asset, what it feeds, and what is exposed when it degrades.

The outcome perspective

KPIs are entities in the same graph, linked by labeled dependencies to the process variables, equipment, operating conditions and events that drive them. Pre-populated KPI groups arrive with AI-suggested dependencies that users review and adjust before the structure is finalised — the graph proposes, the organisation confirms. Deviations, alarms and anomalies are linked automatically to the KPIs they affect as they occur.

When a KPI degrades, impact analysis traverses those dependencies, combining graph structure with data-driven analysis to produce a ranked view of contributing factors, a fishbone of influence, and the critical paths between cause and consequence. Multi-hierarchy views switch the same analysis between plant-wide, unit and shift perspectives — one graph read at different altitudes.

Why one graph beats two

The payoff of a single entity graph is that the perspectives intersect for free. A throughput shortfall resolves, in the same traversal, to the specific assets responsible — with their health scores, open work orders and drawings already attached. A declining health score resolves to the KPIs and production commitments it threatens.

There is no navigation between systems, no export, and no join: the question crosses domains because the edges already do.

Seeing the graph

Cortex makes the graph itself a first-class workspace rather than a hidden substrate. The interactive explorer renders entities as a navigable visual graph: nodes coloured and labeled by UDM type, edges labeled with the relationship they assert, with search, zoom, traversal history and recently visited entities at hand. Filtering by type turns the full graph — thousands of entities across every domain — into exactly the slice a question needs: switch every type off except BOMs and BOM lines, and the product structure of the operation stands alone; add defects and inspections back in, and quality history drapes itself over that structure.

This matters for a reason beyond convenience. Relationships are the product of contextualization, and a structure that can be seen can be challenged: an engineer who notices a wrong or missing edge while exploring is performing graph maintenance, whether they think of it that way or not. The explorer is where the confirmation loop of Section 5 meets its users.

Asking in language, answering in traversals

A single bar serves search and questions alike — find an entity by name, or ask across assets, datasets and documents in plain language. Behind a question, the copilot is explicitly agentic: an orchestrator agent interprets the request and instructs a graph agent, which composes an execution plan before anything runs — the seed entity types to start from, attribute filters that pin the scope to a specific manufacturer, machine class or product variant, the number of relationship hops to traverse, and the entity types the answer may draw on.

Consider a quality engineer asking whether tool wear on a particular manufacturer's punch presses is related to punching defects on one product variant. The plan seeds from tool-wear records and defects, filters machines by manufacturer and type and variants by SKU, walks the graph three hops outward, and scopes the answer to the entity types that matter — machines, variants, panels, punch programs. The traversal crosses maintenance, production and quality domains in one pass, because in the graph they were never separate.

The plan is displayed, step by step, alongside the answer. That is the practical meaning of traceability promised throughout this paper: not a footnote asserting the answer is grounded, but the actual query logic — inspectable, criticisable, and correctable — sitting next to the result it produced.

DESIGN PRINCIPLE: DETERMINISTIC BEFORE PROBABILISTIC

Every relationship in Cortex is either extracted from a source system, inferred and then confirmed, or explicitly asserted by a user.

AI is used to propose relationships and to accelerate their discovery. It is not used to invent them at query time. This is what allows an answer to be traced back to its evidence — and it is the architectural decision that separates a system that can be trusted in a safety review from one that cannot.

4.5 Relationships: proposed by AI, curated by people

The edges of Cortex carry a semantic vocabulary of their own. Built-in relationship types express the recurring structures of industrial reality — `belongsTo` for membership, `partOf` for composition, `inheritsFrom` for classification, `interactsWith` for coupling, `relatedTo` for association — and the vocabulary is open: organisations define custom relationship types to express whatever their operation needs an edge to mean. The relationship layer is as flexible as the entity layer above it.

The types are not interchangeable labels; each supports a different kind of reasoning, and choosing the right one is a modelling decision with consequences.

Relationship type	What it asserts	What it enables
belongsTo	Membership — this cycle belongs to this machine, this machine to this workstation	Hierarchy traversal; rolling condition and history up and down the operation
partOf	Composition — this component is physically part of this assembly	Bill-of-material reasoning; impact of a component failure on the whole
inheritsFrom	Classification — this specific machine is an instance of this equipment class	A new asset inherits the failure modes and monitoring logic of its class the moment it is typed
interactsWith	Coupling — these entities affect each other without containing each other	The cross-hierarchy relationships (shared utilities, heat integration, common tooling) that no tree structure can express
relatedTo	Association — these connect, in a way not yet precisely typed	An honest placeholder: the connection is captured now and refined into a stronger type as understanding grows
Custom types	Whatever the operation needs an edge to mean	Domain semantics the built-in vocabulary does not anticipate — defined by the organisation, first-class in every traversal

Table 4-4. The relationship vocabulary. The type on an edge determines what reasoning that edge supports — which is why relationship curation is a modelling act, not housekeeping.

The lifecycle of an edge runs in both directions. As data lands, AI models propose relationships automatically — from tag structures, drawing connectivity, master-data references, textual mentions and statistical association — so the graph populates at a scale no manual effort could reach. Users then do what only users can: confirm the edges that are right, correct the ones that are wrong, re-type a vague `relatedTo` into a precise `partOf`, and author outright the relationships only experience knows. Every one of those actions is feedback, and none of it is lost — the proposal models learn which relationship types hold where in this plant, and propose better edges next time.

This division of labour is the point. AI provides scale: millions of candidate relationships evaluated as data arrives. People provide judgement: the confirmations and corrections that only operational experience can supply. The graph that results is something neither could build alone — and because the loop never closes, it keeps growing and adapting as long as data flows and people use it. Section 5 examines that learning behaviour as a property in its own right.

4.6 The kinetic graph: from describing to doing

A graph that only describes is a reference work. Operational data architecture increasingly distinguishes two layers a working model of an organisation needs: semantic elements — the entities, properties and labeled relationships this section has described — and kinetic elements: the governed actions through which people and agents actually change the operation. Cortex carries both, and the pairing is what turns a description of the organisation into a working representation of it — a digital twin of the operation in the full sense, covering not only what exists but what can be done.

An action, in this sense, is a governed transaction: it modifies one or more entities under defined logic, with declared parameters, validation rules and permissions, rather than as an ad-hoc edit. Creating a work order, updating a health score, recommending a setpoint change, initiating an inspection, escalating to a supervisor — each is an action type with a defined shape, and each reaches the real world by writing back to the systems of record through their own interfaces. The CMMS, MES and historian remain authoritative for what they own; the graph orchestrates the change and records that it happened.

The semantic and kinetic layers of Cortex

The semantic layer knows the operation; the kinetic layer changes it - and every change becomes knowledge

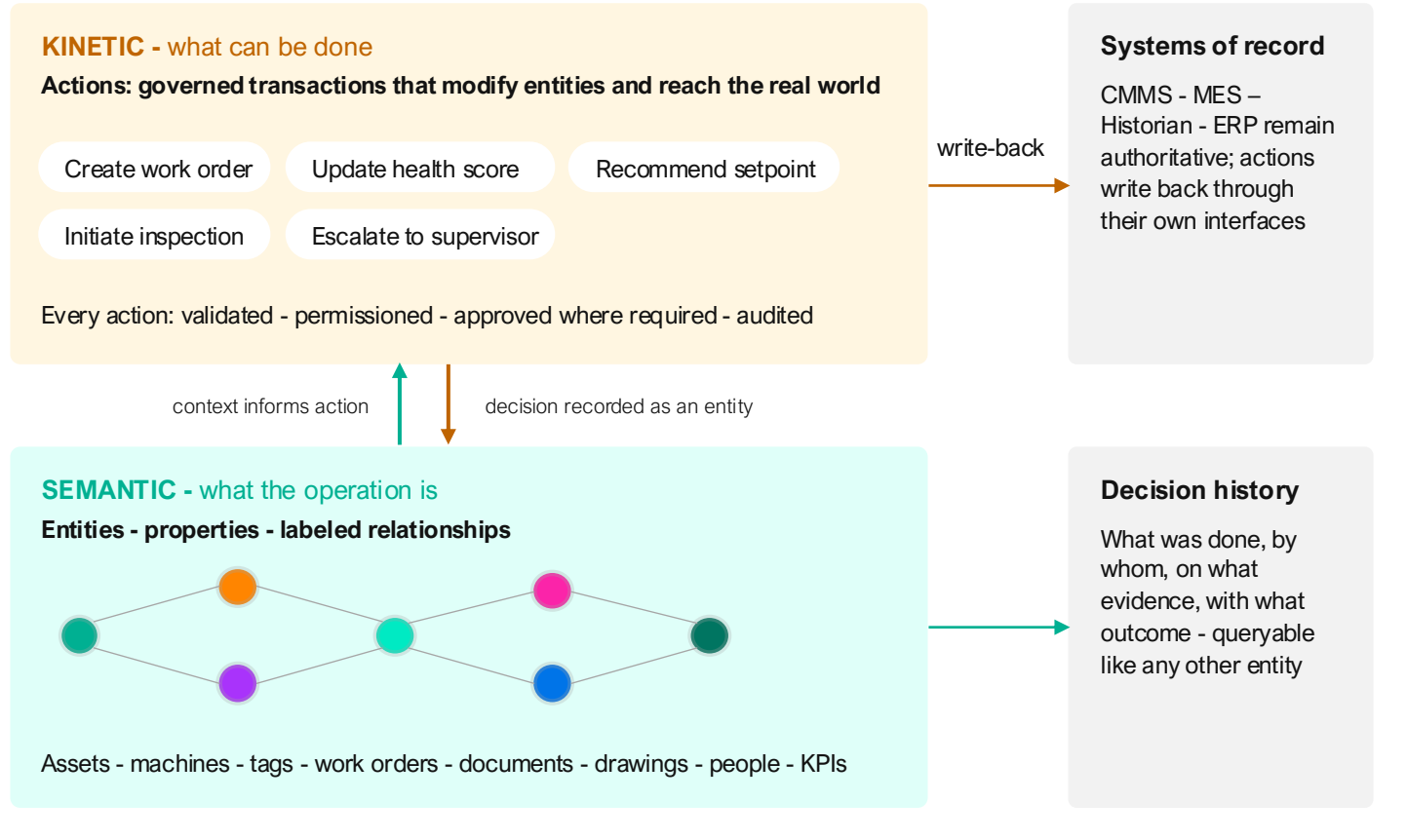


Figure 4-2. The semantic layer knows the operation; the kinetic layer changes it. Actions write back to the systems of record — and are themselves recorded as entities, so the decision history becomes part of the graph.

The consequence worth dwelling on is the last one: decisions become data. Every action taken through the graph is itself recorded as an entity — what was done, by whom, on what evidence, with what outcome — linked to the assets, alarms and analyses that prompted it. Closing the gap between insight and execution also closes the gap between execution and memory. An operation's decision history becomes queryable exactly like its equipment history: the next engineer facing the 03:40 alarm can ask not only what the graph knows about the compressor, but what was done the last three times, and whether it worked.

Because actions change the real operation, they inherit the strictest controls in the platform: validation before execution, permissioning by role and scope, approval workflows where the organisation requires them, and a complete audit trail — with the graduated autonomy model of Section 8 determining whether a given action may be taken by an agent alone, an agent with approval, or a person. Section 9 details the control set; the point here is architectural. The kinetic layer is not a side door around governance. It is where governance becomes enforceable, because every change flows through one typed, logged, permissioned channel.

05

The Neural Graph: A Structure That Learns

Everything described so far — entities, labeled edges, canonical models — could in principle be built as a static artefact: mapped once, deployed, and left to age. What earns Cortex the "semantic brain" framing of this paper's title is that it is not static. The graph behaves less like a map and more like neural tissue: the pathways that are used strengthen, the structure adapts to what the operation actually does, and the whole becomes more capable with exercise rather than less.

This is not a metaphor stretched over ordinary software. It describes three specific, reinforcing mechanisms.

The neural loop - how Cortex learns

Five stages, one cycle;
every turn leaves the
graph more accurate
than the last

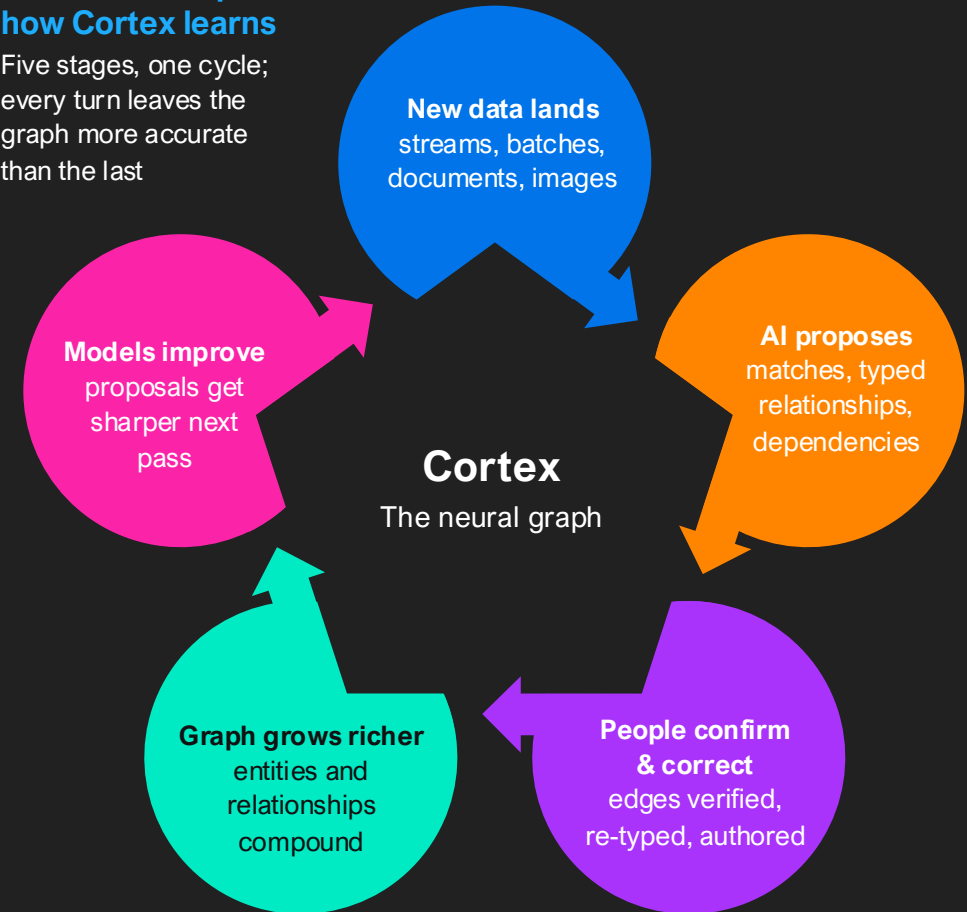


Figure 5-1. The neural loop. Each turn of the cycle is small; the compounding across turns is what separates a learning graph from a static integration.

5.1 Reliability compounds with use

The reliability of the graph is a function of how many people use, draw on and enrich it. Every query exercised against the graph, every relationship a user confirms, every correction an engineer makes when a proposed match is wrong — each of these is retained and feeds the structure. Data that is worked with becomes better data; better data yields insights people trust; trusted insights pull more of the organisation onto the graph; and a larger, more demanding user base enriches it faster still.

The dynamic is the same one that governs compounding interest, and it runs equally in reverse for the alternatives: a point integration is at its best on the day it ships and decays with every plant change thereafter. The graph inverts that curve. It is at its weakest on day one and appreciates from there.

5.2 The models improve the contextualization that feeds them

The AI beneath the graph is not a fixed component either. The models that propose entity matches and typed relationships, extract drawing connectivity, classify documents and suggest KPI dependencies adapt and improve as evidence accumulates — continuously refining the contextualization process

itself, sharpening the accuracy and relevance of what is extracted, and enriching the graph with relationships that earlier passes could not see.

The loop is worth stating precisely, because it is where the compounding actually happens: the graph supplies context that makes the models better; the improved models contextualize new data more accurately; the more accurately contextualized data makes the graph richer; and the richer graph supplies still better context. Each turn of that cycle is small. Three years of it is transformative.

5.3 The learning loops, concretely

The abstract cycle above is implemented as specific feedback paths, each one already described in this paper and each one closing through the graph:

Capability	What feeds back	What improves
Entity matching	Every human confirmation or correction of a proposed match, retained as training signal	The matching pipeline learns the plant's naming conventions; reconciliation of each subsequent scope gets faster and more confident
Relationship proposal	Confirmations, corrections and re-typings of AI-proposed edges, plus relationships users author directly	The proposal models learn which relationship types hold where; the vocabulary and the graph grow together as new data lands
P&ID extraction	User-verified bounding of unrecognised elements and corrected connections	Extraction accuracy on the remaining drawing library, and on every future revision ingested
Vision AI	Production images, detections and operator annotations, governed as training and test data; drift triggers on lighting, optics and material changes	Detection accuracy compounds from operations; models are retrained before drift becomes failure
KPI dependency modelling	Accumulating operational evidence across cycles and upsets	Machine learning surfaces dependency patterns among KPIs and process variables that no one thought to assert
Agent operations	Outcomes of agent recommendations and actions, logged and auditable	Agents learn from what worked; autonomy expands on evidence rather than optimism

Table 5-1. Five feedback paths, one destination. Each loop closes through the graph, so learning anywhere becomes context everywhere.

The kinetic layer of Section 4.6 adds the deepest loop of all: decisions themselves. When every action taken through the graph is recorded as an entity — what was done, on what evidence, with what outcome — the graph learns not only what the operation is, but which interventions actually work in it. That is knowledge no source system holds today, because today it evaporates the moment the shift ends.

THE COMPOUNDING PROPERTY

Data in a well-maintained industrial knowledge graph behaves like compounding interest. More use produces more enrichment. More enrichment produces higher-quality relationships. Higher-quality relationships produce more trusted answers. More trusted answers produce broader adoption — which produces more use.

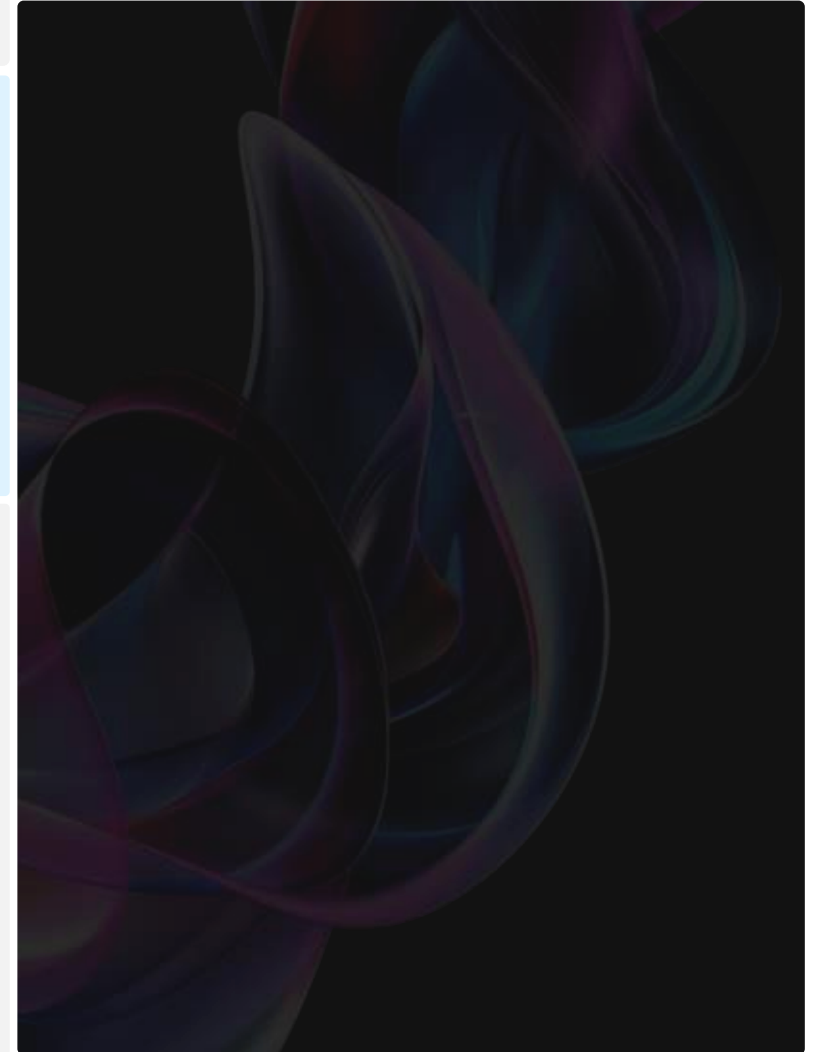
The corollary is that the first six months of a graph programme feel slower than a point integration, and the third year is not comparable to one at all.

5.4 What a learning graph changes strategically

A structure that improves with use inverts the economics of every previous approach to industrial data. Integration projects are budgeted as recurring costs because their output decays: the organisation pays for the same reconciliation on a cycle, forever. A neural graph converts that recurring cost into an appreciating asset — the digital representation converges toward the plant instead of drifting away from it, and the cost of the next question, the next use case and the next site falls with every one delivered.

It also changes who the graph is for. A static model serves the people who commissioned it. A learning graph serves everyone whose use improves it — which means the operator's correction at 03:40 and the reliability engineer's confirmed match are not overhead on the system. They are the system, accruing to every colleague who asks the next question.

**A map is drawn once and begins to decay.
A brain is used, and improves.**



06

Building and Maintaining the Graph

A knowledge graph is only as credible as its construction process. This section describes how IRIS Foundry builds one from an existing industrial estate, and — more importantly — how it keeps it accurate afterwards.

6.1 Ingestion

IRIS Foundry ships more than 150 pre-built industrial connectors covering the OT and IT systems found in asset-intensive operations. Native protocol support includes OPC UA and OPC DA, MQTT, Modbus and REST, with secure data streaming. Process historian coverage includes AVEVA PI, GE Vernova Proficy, Honeywell Forge, Siemens SIMATIC and Rockwell FactoryTalk. Enterprise coverage includes SAP, IBM Maximo and comparable CMMS and ERP platforms, alongside SCADA, DCS, MES, LIMS and quality systems.

Ingestion handles structured, semi-structured and unstructured data on equal terms. Engineering files, P&IDs, inspection imagery, 3D models, point clouds and free-text maintenance notes are first-class inputs rather than attachments, because a substantial share of the relationships worth capturing exist only in those artefacts.

6.2 P&ID ingestion and tag reconciliation

Process and instrumentation diagrams are the densest source of relational truth in most plants and the least machine-readable. IRIS Foundry applies AI-assisted extraction to recover assets and sensors from uploaded drawings, infer connections between them, map assets and sensors to drawing numbers and plant systems, follow connected assets and tags across multiple drawings, and capture drawing metadata including plant system and subsystem.

Extraction is explicitly review-assisted rather than autonomous. Users bound and define elements the parser did not identify, and verify or correct proposed connections and mappings. This is deliberate: a wrongly inferred connection in a P&ID propagates into every downstream answer, so the design accepts a human confirmation step at the point where an error would be most expensive.

Tag reconciliation runs alongside, mapping identifiers across historians, DCS and ERP systems, resolving naming inconsistencies, and producing a canonical asset hierarchy. Where a maintenance system holds equipment master records, functional locations and BOM structures, these seed the hierarchy — including where that data is only partially populated, which is the normal condition rather than the exception.

Stage	Process	Detail
1 Ingest	Data acquisition	Historian, OPC-UA/SCADA, ERP, MES and additional sources identified in scoping. Real-time and batch.
2 Cleanse	Data quality	Gap detection, outlier removal, unit normalisation, tag reconciliation. Leverages existing historian asset-framework metadata where present.
3 Contextualize	Asset mapping	Asset hierarchy built from historian asset framework and ERP equipment master. P&ID linkage. Knowledge graph enrichment.
4 Unify	UNS / UDM build	ISA-95 aligned unified namespace; a single data model across the scoped operations. Foundation for multi-site scale-up.
5 Analyze	AI/ML processing	Anomaly detection, cause identification, risk scoring, remaining-useful-life estimation.
6 Act	Operational response	Application alerts, copilot recommendations, CMMS write-back, value dashboard updates.

Table 6-1. The six-stage pipeline from raw acquisition to operational action. Stages three and four are where contextualization occurs and where the durable value is created.

6.3 Scale-up economics

The first unit is the expensive one. Tag reconciliation, asset mapping and graph construction for an initial scope is measured in weeks. The same work at the second and subsequent sites is measured in days, because the namespace and data model templates are established and each new site inherits them, requiring site-specific fine-tuning rather than fresh modelling.

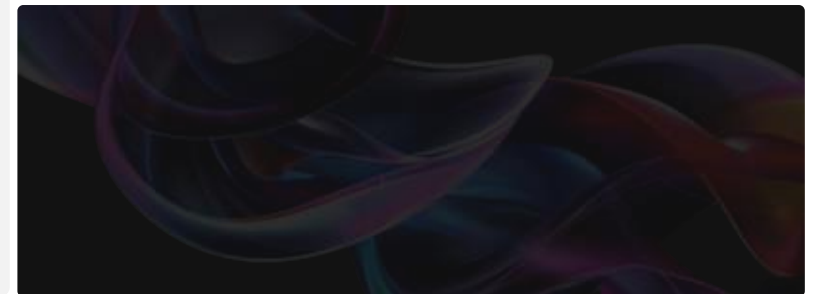
This asymmetry has a direct bearing on programme design. An organisation that pilots at one site and evaluates on the cost and duration of that pilot will systematically underestimate the return of the programme, because it is measuring the least efficient instance it will ever build.

The first site pays for the model. Every subsequent site inherits it.

6.4 Keeping the graph current

Maintenance of the graph is continuous and largely automatic. New data is contextualized against the existing model on arrival. Changes in source systems — a new functional location, a revised procedure, a replaced asset — propagate as updates rather than requiring a rebuild. Machine learning identifies new dependency patterns among KPIs and process variables as evidence accumulates. Corrections made by engineers are retained and applied.

The governance question this raises — who is permitted to assert or change a relationship — is addressed in Section 9. It is worth stating here that the answer is not "anyone", and that the audit trail of relationship changes is treated with the same seriousness as the audit trail of data access.



07

The Industrial LLM: Language Grounded in the Graph

A knowledge graph makes relationships machine-readable. It does not, by itself, make them accessible to someone who does not write queries. That is the role of the language layer — and the role is only safe when the language layer is subordinate to the graph rather than independent of it.

7.1 Why general-purpose models are insufficient alone

Large language models trained on general text have no knowledge of a specific facility. Asked about a named compressor, such a model will produce a fluent answer drawn from patterns in its training data, with no mechanism for knowing whether it applies to this machine, this configuration or this revision of the procedure. The generative capability that makes these models useful is precisely what makes them unreliable when facts must be specific: they are trained to produce plausible continuations, and plausibility is not accuracy.

In industrial settings this failure mode is not merely inconvenient. A confidently wrong answer about a pressure limit, an isolation procedure or a permitted operating envelope is a safety exposure. The correct architectural response is not to suppress generation but to constrain what the model is permitted to reason over.

7.2 Domain training

IRIS Foundry is powered by SymphonyAI's Industrial LLM, trained not on generic internet text but on the vocabulary, semantics and logic of industrial operations — engineering schematics, equipment manuals, P&IDs, FMEAs, procedures, maintenance logs, shift handover notes and process documentation from asset-intensive industries.

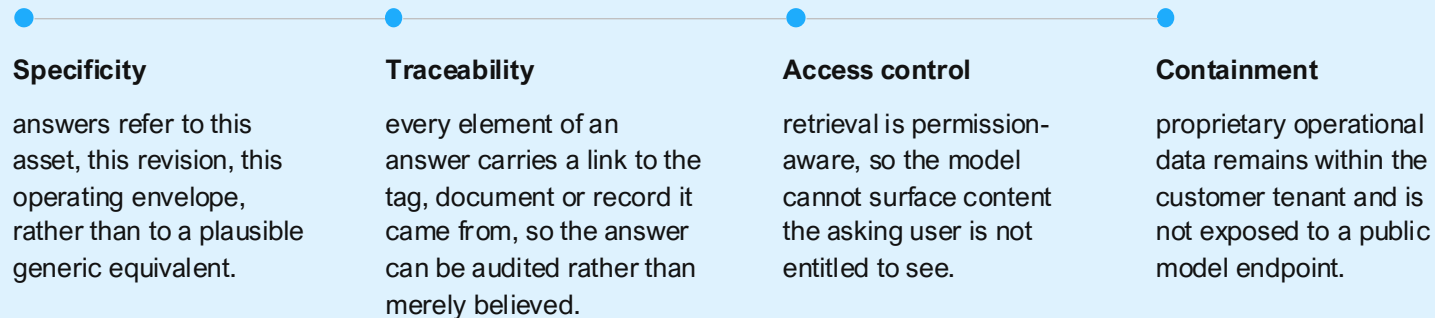
Domain training changes what the model understands before any retrieval occurs. It knows that a shift handover note is a different kind of evidence from a vendor manual. It parses the abbreviations, unit conventions and elliptical phrasing of maintenance records. It recognises the structure of an FMEA. These are not marginal gains; they determine whether retrieval returns the right passage at all.

7.3 Grounding: the reasoning engine, not the generative engine

The architectural principle is to use the reasoning capability of the model while supplying the facts externally. Retrieval-augmented generation over Cortex provides contextualized industrial data directly to the model as the specific content it must use in formulating a response, rather than relying on the model to produce an answer from its training distribution.

The measured effect of graph grounding is significant. Published research on combining knowledge graphs with retrieval-augmented generation reports hallucination reductions of approximately 73% against an ungrounded GPT-4o baseline, compared with roughly 63% for retrieval-augmented generation without a graph — indicating that the graph structure contributes materially beyond retrieval alone.

WHAT GROUNDING BUYS, PRECISELY



7.4 Deployment flexibility

The platform is model-agnostic. In cloud deployments, managed model services are consumed over private connectivity with public network access disabled. In on-premise deployments, external model APIs are optional and reached over a controlled outbound path; in fully air-gapped configurations they are removed entirely and replaced by a self-hosted open-weight model. The trade-off is explicit — an air-gapped configuration forgoes cloud-dependent generative features — and the decision is the customer's to make against their own data residency and sovereignty requirements.

7.5 Where language meets the workflow

Access to the language layer is not confined to a dedicated application. Through the Foundry MCP interface, IRIS Foundry integrates into Microsoft Teams and Microsoft 365 Copilot, allowing plant teams to reach live asset health, process data and agent recommendations inside the collaboration tools they already use. The design intent is that context reaches the user where the work happens rather than requiring a separate destination.

08

From Context to Action: Agentic AI

A contextualized graph and a grounded language layer produce good answers. They do not, by themselves, produce outcomes. The last mile — insight becoming action — is where most industrial AI programmes stall.

IRIS Flows is the agentic execution layer built on IRIS Foundry: an environment where intelligent agents are created, deployed, orchestrated and governed, operating on the contextual understanding the graph provides.

8.1 Teams rather than tasks

Conventional automation encodes tasks: if X occurs, do Y. Industrial reality defeats this, because events rarely follow linear patterns — root causes evolve, conditions shift, and outcomes depend on multi-variable dependencies.

IRIS Flows assigns each agent a responsibility rather than a fixed instruction. One agent tracks the health of a rotating asset class; another evaluates energy consumption patterns; a third interprets work order data; a fourth guides technicians through repair procedures. Agents communicate, delegate, escalate and cooperate, mirroring human coordination with machine consistency. Intelligence is distributed across specialised agents rather than centralised, which increases resilience, reduces bottlenecks and supports parallel problem-solving.

Agent family	Responsibility
Reliability and maintenance	Asset health tracking, failure prediction, maintenance planning support
Production and process optimization	Throughput, yield, energy and process-condition optimisation
Safety, compliance and environmental	Condition monitoring against safety envelopes and regulatory limits
Alarm management and event intelligence	Alarm rationalisation, deduplication, prioritisation and correlation
Connected worker and procedure automation	Dynamic procedure delivery and frontline execution guidance
Knowledge, documentation and retrieval	Locating, summarising and contextualising technical content on demand

Table 8-1. The IRIS Flows agent ecosystem. Each family holds a standing responsibility rather than a scripted trigger.

8.2 Graduated autonomy

Not every agent should act autonomously, and no organisation is ready for high autonomy on day one. IRIS Flows supports three autonomy models, selected per agent to match organisational readiness, regulatory requirements and safety constraints.

Mode	Agent behaviour	Appropriate for
Assistive	Monitors, analyses and recommends; requires human approval before any action	Early deployments and high-risk decisions
Augmented	Executes automatically within predefined boundaries; humans intervene when the agent flags ambiguity or risk	Non-critical operations with well-understood limits
Autonomous	Diagnoses, acts, updates systems and coordinates workflows within defined constraints	Routine or low-risk processes with established track record

Table 8-2. Graduated autonomy allows trust to be earned incrementally and evidenced, rather than assumed at deployment.

The graduation is the point. An organisation validates agent performance in assistive mode against decisions its experts would have made, expands to augmented mode where the record supports it, and reserves autonomous mode for processes where the risk profile and the evidence both justify it.

8.3 Closing the loop

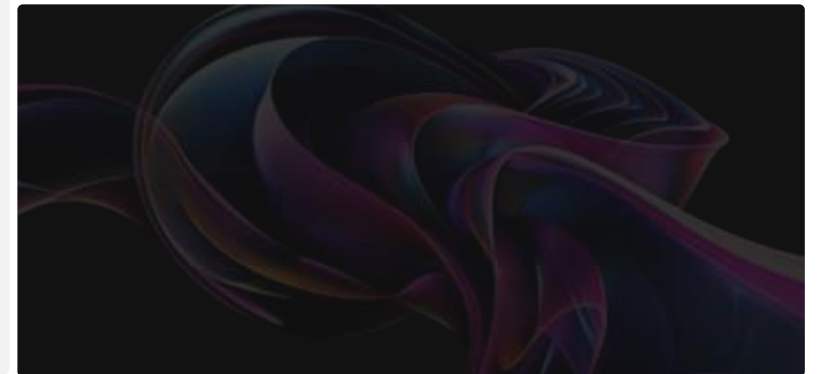
Agents execute workflows spanning OT, IT and frontline teams. They can trigger work orders in maintenance systems, update asset health scores, modify MES or historian records based on validated observations, send instructions through collaboration tools and mobile devices, initiate inspections, alert supervisors when safety risks are detected, and generate reports and log learnings automatically.

Integration with Proceedix Connected Worker extends execution to the physical world: step-by-step instructions, augmented-reality overlays and procedure-driven guidance delivered to frontline technicians, with dynamic procedures tailored to the asset's current condition, natural-language answers to questions asked in the field, and defined escalation paths when observations diverge from expectation. This creates a closed loop from digital detection to physical execution.

Agents do not replace the workforce. They remove the retrieval work that consumes it, and they make the resulting decisions consistent.

8.4 Building without engineering

IRIS Forge, the no-code application environment, allows users to describe an objective in natural language and have a data-connected, context-aware application generated against the graph. Engineers, operators, analysts and plant managers can produce persona-specific applications — dashboards, workflows, decision support — without a development cycle. The applications inherit the graph's context and access controls by construction rather than by configuration.



09

Governance, Security and Operational Trust

Contextualized data and autonomous agents both expand what a system can reach and what it can do. In regulated and safety-critical environments, that expansion is only acceptable when it is accompanied by controls proportionate to it. This section describes the control set.

9.1 Architectural posture

IRIS Foundry follows a defense-in-depth model across network, identity, data and compliance. Network zones follow the Purdue / ISA-95 model. The platform holds no agents in the control zone and performs no control writes — a deliberate architectural boundary that keeps the platform out of the safety-instrumented path regardless of how autonomy is configured above it. Every access is treated as untrusted until authenticated and authorised.

Services are containerised microservices orchestrated by Kubernetes, communicating through documented, secured APIs, and deployable identically to private cloud or on-premise. The deployment decision is therefore a placement decision rather than an architecture change — the same control set applies in either case.

9.2 Identity and access

Identity is managed through an internally hosted identity broker deployed inside the secure cluster, federating upstream to the corporate identity provider over SAML 2.0 or OIDC. Authentication occurs against the customer's source of truth without local password storage, and external assertions are translated into short-lived, cryptographically signed tokens scoped to the platform.

- **Role-based aggregation** — granular permissions bundled into reusable roles across three core actions: read, write and configure.
- **Centralised assignment** — administrators assign predefined roles to users and service accounts, simplifying onboarding and compliance audit.
- **Scoped authorization** — scopes dynamically restrict where actions may occur, enforcing isolation across data boundaries. MFA is enforced for privileged and remote access.

Permission awareness extends into retrieval. Document indexing is permission-aware, so the copilot and agents cannot surface content the asking user is not entitled to see — a property that must be built into the retrieval layer rather than filtered afterwards.

9.3 Data protection

Data at rest is encrypted using customer-managed encryption keys via the cloud key-management service, or volume and disk-level encryption on-premise, so underlying disks, snapshots and backups remain unreadable without explicit key authorisation. Transport uses TLS 1.3; unencrypted links are dropped at the network layer. In cloud deployments, worker nodes carry zero public-IP exposure.

9.4 Agent governance

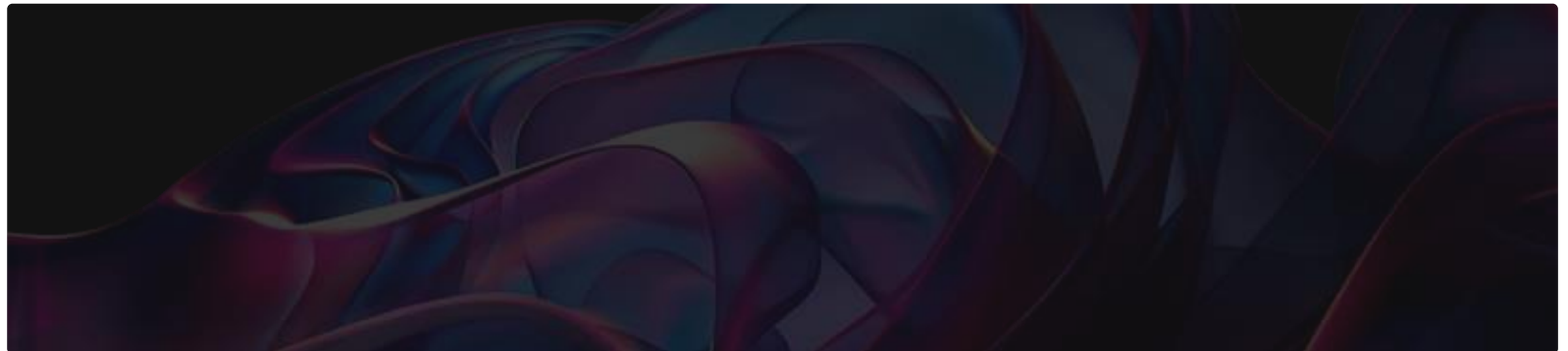
Agent actions are governed by explicit rules, guardrails, approval workflows and auditability. Organisations retain control over which systems agents may modify, which action types require approval, which thresholds trigger escalation, who is notified when interventions occur, and how autonomous each agent is permitted to be. Allowlists and denylists are explicit; safety envelopes constrain decisions to acceptable operating limits; human override remains available at all times.

Every agent action is logged and auditable, and agents must justify their recommendations so that operators understand the rationale rather than receiving a verdict. This requirement — that the reasoning be inspectable, not merely the outcome — is what makes graduated autonomy defensible to a regulator or a safety case.

9.5 Assurance

Control area	Practice
Standards alignment	ISO 27001:2022, NIST CSF, SOC 2 Type II and OWASP Top 10 controls; certified controls inherited from underlying cloud infrastructure
Testing	Vulnerability assessment and penetration testing twice yearly; container images scanned automatically in the CI/CD pipeline and registry
Scope of testing	Custom workflows, automation pipelines and integration endpoints, including prompt-injection resistance; ingestion paths confirmed isolated from plant networks
Patch management	Defined cadence; on-premise packages distributed internally, removing worker-node internet dependency
Software lifecycle	Requirements, design, review, security testing, functional testing, deployment, monitoring and retirement, with static analysis and dependency review embedded
Availability	99.9% SLA; horizontal scaling, load balancing and distributed deployment across plants, edge clusters and redundant cloud environments

Table 9-1. Platform assurance practices. Prompt-injection resistance is included in penetration testing scope, reflecting that the language layer is an attack surface.



10

Use Cases in Practice

The value of contextualization is best demonstrated against specific operational questions. Each case below follows the same structure: the question, why it is hard without a graph, and what the graph changes.

10.1 Root cause analysis

The question: why did this KPI degrade, and what do we do about it?

Without a graph, this is the seven-step manual cycle of Section 2 — hours to days, non-reusable, dependent on the investigator's tenure. With Cortex, KPI dependencies are already modelled as labeled edges: impact analysis traverses the graph to rank contributing factors, produces a fishbone view of influence, and identifies the critical paths. The investigator moves directly from a ranked hypothesis set to verification, and the traversal is inspectable at every hop.

10.2 Predictive maintenance and asset reliability

The question: which assets are heading for failure, and how long do we have?

Predictive models are only as good as the context available to them. Production state from an MES tells a model whether equipment is at full load, ramping or changing blend — which materially reduces false positives from normal transients. Maintenance history tells it what has already been replaced. Failure-mode libraries tell it what patterns to expect for this asset class. The graph supplies all three to the model as features rather than requiring a data scientist to assemble them per project.

IRIS Foundry combines this with machine reasoning engines incorporating machine and process physics to determine probable root causes and mitigating actions, and a pre-built asset model library with predefined failure-mode patterns covering more than 85% of common industrial assets.

10.3 Alarm rationalization

The question: which of these 400 alarms matters?

Alarm floods are a contextualization failure by definition — each alarm is locally valid and collectively uninterpretable. Graph relationships allow correlated alarms arising from a single upstream cause to be grouped, deduplicated and prioritised by consequence rather than by severity flag, with the causal chain presented alongside.

10.4 Knowledge retrieval and workforce continuity

The question: what does the person who left know that I need?

This is the case with the largest long-term exposure and the least visible short-term urgency. Contextualized retrieval over procedures, manuals, maintenance history, inspection records and shift handover notes converts an oral tradition into a queryable one. It does not fully replace experiential judgement — but it removes the dependency on a specific individual being reachable at 03:40.

10.5 Cross-site model transfer

The question: this worked at one site — how do we get it everywhere?

Standards-aligned namespace and data model mean an anomaly detection model built for an asset class at one site applies to comparable assets elsewhere with reconfiguration rather than reconstruction. This is the mechanism by which organisations exit the pilot-purgatory cycle: the constraint on scale-up becomes site onboarding rather than per-site model development.

10.6 Sustainability and energy

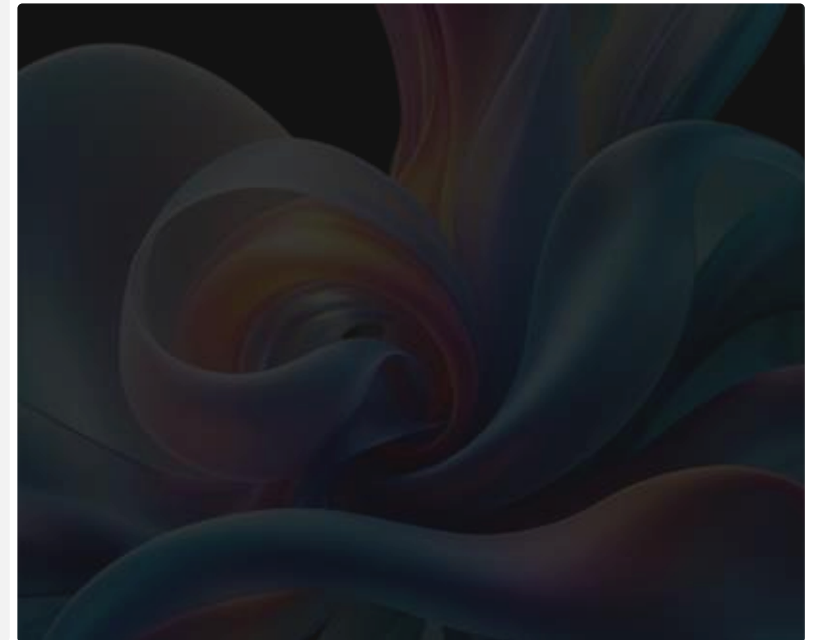
The question: where is energy being wasted, and what does fixing it cost?

Energy intensity is a cross-domain question by nature — it depends on equipment condition, process configuration, production mix and ambient conditions simultaneously. Cortex models these dependencies explicitly, allowing energy KPIs to be decomposed into contributing factors and traded off against throughput and reliability objectives rather than optimised in isolation.

A NOTE ON SEQUENCING

These use cases are listed in approximate order of implementation difficulty, not value. Most successful programmes begin with root cause analysis or predictive maintenance on a bounded scope, because both produce evidence quickly and both exercise the graph in ways that expose modelling gaps early.

Workforce continuity and cross-site transfer deliver the largest structural returns but are difficult to justify as an opening use case, because their value accrues over a longer horizon than a typical pilot evaluation window.



11

Deployment and Adoption

11.1 Deployment models

IRIS Foundry supports three deployment models. All three run the same containerised architecture, so the choice is driven by data residency, sovereignty and network policy rather than by capability.

Model	Description	Best suited to	Typical go-live
SaaS cloud	Hosted on SymphonyAI managed cloud infrastructure	Pilots, rapid deployments, flexible data residency requirements	2–4 weeks
Private cloud / VPC	Deployed within the customer's own cloud tenancy; full data sovereignty	Regulated industries; strict data governance or compliance mandates	4–8 weeks
On-premise / air-gapped	Fully local deployment; no external connectivity required; supports air-gapped OT networks	Critical infrastructure with strict isolation requirements	8–12 weeks

Table 11-1. Deployment options. Go-live figures reflect platform deployment; use-case delivery timelines depend on scope.

11.2 A phased adoption model

Evidence-led adoption is the pattern that succeeds: validate on a defined initial scope, then scale what is proven, so enterprise commitment follows demonstrated value rather than preceding it.

PHASE 0**Scoping**

Source-system inventory, access provisioning, success criteria and baseline establishment. The most consequential activity is agreeing baselines: value claims made later are only defensible against numbers agreed before the work started.

PHASE 1**Foundation and first use cases**

Data fabric construction for a bounded scope: P&ID ingestion and tag reconciliation, unified namespace and data model deployment, Cortex entity graph construction linking assets, sensors, failure modes, work orders, P&ID nodes and operational events, KPI dependency trees with impact analysis models, and an open API layer exposing the contextualized model. Two or three use cases are delivered against this foundation.

PHASE 2**Scale**

Additional units and sites inherit the namespace and data model. Use-case backlog expands. Agentic workflows are introduced in assistive mode. Applications are built by customer teams in the no-code environment rather than by the vendor.

PHASE 3**Autonomy and self-sufficiency**

Agent autonomy expands where the operating record supports it. Customer teams create and govern their own agents and applications. The programme's dependency on external delivery capacity declines.

11.3 Enablement

Adoption depends on capability transfer as much as on technology. Effective programmes combine role-specific bootcamps using real or anonymised operational data, self-paced modules covering architecture, ingestion, contextualization and agent design, on-demand reference material, and tracked adoption metrics — completion rates, feature adoption and usage trends — so that enablement is measured rather than assumed.

The objective is a core group of internal champions capable of building without vendor involvement. Programmes that omit this step typically plateau at the use cases the vendor delivered.



12

Measuring Value

Contextualization is an infrastructure investment that pays through the use cases built on it. That makes measurement a design problem: value must be attributed to the use case while the foundation is credited for enabling it.

12.1 The value bridge

Reliability and process outcomes translate to financial results through four principal drivers.

Driver	Typical range	Mechanism
Downtime reduction	30–50%	Avoided unplanned outages and lost production
Maintenance cost	10–40% reduction	Less emergency labour, spares consumption and over-servicing
Asset life extension	20–40%	Deferred capital replacement
Safety and integrity	Fewer incidents	Earlier detection; compliance evidence generated as a by-product

Table 12-1. The value-driver tree. Ranges reflect outcomes across SymphonyAI deployments and published industry analysis; site-specific results depend on baseline maturity, asset criticality and production economics.

12.2 Reported outcomes

Across deployments, customers have reported the following in the three-to-twelve-month window following implementation.

- **Unplanned downtime** — reductions of up to 30–50% through predictive maintenance and anomaly detection, with 15–25% reduction observed within the first 90 days of deployment.
- **Production efficiency** — throughput increases of 5–10% through optimised operations and process recommendations.
- **Return on investment** — 30–300% within the first year depending on use case; 3–6× ROI documented in year one for reliability-led programmes.
- **Time to value** — new analytics use cases deployed in weeks rather than months, against traditional industrial analytics project timelines.
- **Specific programmes** — a predictive reliability model for a global chemical manufacturer reduced downtime by 15%; digital twin-based monitoring across more than 10,000 assets for an energy company improved asset life by 25%.

1,000+

plants deployed

30+

countries

97%+

customer retention

150+

pre-built connectors

Figure 12-1. IRIS Foundry deployment footprint across chemicals, oil and gas, energy, semiconductor, pharmaceuticals, utilities, mining and metals, food and beverage, and CPG.

12.3 Measuring the foundation, not only the use case

Use-case metrics understate a contextualization programme, because they capture the return on the first application and not the reduced cost of every subsequent one. Three foundation metrics are worth tracking alongside financial outcomes:

1. Time-to-answer for cross-domain questions. Measure the elapsed time from question to evidenced answer for a standard set of investigations, before and after. This is the most direct measure of what the graph changed.
2. Cost of the nth use case. Track implementation effort per use case over time. A healthy graph programme shows a declining curve; a flat curve indicates the foundation is not being reused and is functioning as a series of point integrations.
3. Breadth of direct access. Measure how many distinct people query operational data directly without an intermediary. This is the metric most closely tied to workforce-continuity exposure, and the one most often left unmeasured.

A programme that improves financial KPIs while these three remain static has purchased a set of applications, not a foundation.

13

In Summary

The industrial data problem was never a volume problem. Organisations that instrumented aggressively arrived at the same place as those that did not: an estate full of accurate, well-maintained, mutually unintelligible records, and a small number of experienced people holding the relationships that make them usable.

Contextualization addresses that directly. A semantic industrial knowledge graph captures the relationships that exist in the plant but are recorded in no system — what feeds what, what fails how, what governs which procedure, what moves which KPI — and maintains them as the plant changes. That structure is what allows a question to be answered without a human performing the retrieval, and what allows an AI system to reason about a specific facility rather than a plausible generic one.

IRIS Foundry implements this as Cortex: an open, flexible, labeled entity graph over a medallion data progression that runs from immutable raw ingestion through canonical contextualization to perspective-specific projections; a structure that learns, growing more reliable the more it is used and enriched; a domain-trained Industrial LLM grounded in that graph; an agentic execution layer with graduated autonomy and explicit governance; and a deployment model that runs identically in managed cloud, customer tenancy or air-gapped isolation.

Your data already holds the answers. Context is what lets the whole workforce — and the agents working alongside them — ask the right question and get it answered in time to matter.

Where to start

The evidence-led pattern is consistent across successful programmes. Choose a bounded scope where the context gap is expensive and measurable — typically a unit or asset class where root cause is slow. Agree baselines before starting. Build the foundation properly for that scope rather than shortcutting to the application. Validate the traced answers against what your own experts would have concluded. Then extend, and expect the second site to cost a fraction of the first.

The foundation compounds. Every asset, procedure, failure mode and event added to the graph makes the next question cheaper to answer than the last — which is the property that separates an infrastructure investment from a project.



www.symphonyai.com | 

To discuss a contextualization assessment for your operation, contact SymphonyAI Industrial at symphonyai.com/industrial.



Appendix A

Glossary

Term	Definition
Action	A governed transaction through which a person or agent modifies entities — validated, permissioned, audited, and written back to the systems of record. The unit of the kinetic layer.
Contextualization	The automated discovery and maintenance of relationships between data elements that are connected in the physical world but not explicitly represented in any source system.
Cortex	IRIS Foundry's knowledge graph — an open, flexible, labeled entity graph in which every asset, sensor, process, event, work order, document and KPI is a node and every relationship is a labeled, traversable edge.
Digital twin	The aggregation of all data types and datasets, historical and real-time, related to a physical asset or set of assets, linked in a way that mirrors the real world.
Entity matching	The evidenced establishment that records held in different systems refer to the same physical object — the keystone step on which every cross-system relationship depends.
FMEA	Failure Mode and Effects Analysis — a structured method for identifying how equipment fails and with what consequence.
Industrial DataOps	The ingest, transformation, contextualization and governance architecture supporting industrial data at operational scale.
Industrial LLM	A large language model trained on industrial content — schematics, manuals, P&IDs, FMEAs, procedures and maintenance records — rather than generic text.
IRIS Flows	The agentic execution layer where industrial agents are created, deployed, orchestrated and governed.
IRIS Forge	The no-code, prompt-driven application environment for building context-aware industrial applications.
ISA-95 / ISA-88	International standards defining enterprise-control integration hierarchy and batch process control models respectively.
Kinetic layer	The action side of the graph — what can be done to the operation — as distinct from the semantic layer, which describes what the operation is.
Medallion architecture	The Bronze (raw), Silver (canonical) and Gold (application-optimised) data layering pattern.
Neural graph	The learning behaviour of the Cortex entity graph: relationships, models and extraction pipelines that improve continuously through use, feedback and accumulating operational evidence.
P&ID	Process and Instrumentation Diagram — the engineering drawing describing process equipment, instrumentation and connectivity.
Purdue model	A reference architecture defining network segmentation levels between enterprise IT and industrial control systems.
RAG	Retrieval-Augmented Generation — supplying a language model with retrieved source content to ground its response in specific facts.
Relationship type	The semantic label an edge carries — belongsTo, partOf, inheritsFrom, interactsWith, relatedTo, or a custom type the organisation defines. The type determines what reasoning the edge supports.
RUL	Remaining Useful Life — the estimated operating time before an asset requires maintenance or replacement.
Source Data Model	Data as its system of record shaped it, ingested immutably in its original state at the Bronze layer — the basis of lineage, auditability and re-derivation.
Unified Data Model (UDM)	The canonical, standards-aligned schema set defining how assets, sensors, failure modes, work orders and production context relate, independent of source system. The representation applications and agents consume.
Unified Namespace (UNS)	A single consistent hierarchy mapping every sensor tag, asset, process area and data source into one addressing structure.



Appendix B

Sources and Notes

External sources

1. Fluke Corporation, "Unplanned Downtime Costs Manufacturers Up to \$852M Weekly," October 2025. Survey of 600+ senior decision-makers and maintenance professionals across the United States, United Kingdom and Germany, spanning food and beverage, oil and gas, life sciences and automotive. Source of the \$852M weekly, \$1.7M per hour, 61% incidence and weekly-frequency figures cited in Sections 1 and ES.
2. Gartner, "Data Literacy: A Guide to Building a Data-Literate Organization." Source of the data literacy definition — "the ability to read, write and communicate data in context, with an understanding of data sources and constructs, analytical methods and AI techniques" — and of the finding that poor data literacy ranks among the top five obstacles to data and analytics success in the 2024 CDAO Survey.
3. Deloitte and The Manufacturing Institute, manufacturing skills gap research. Source of the 2.4 million unfilled roles by 2028, 2.6 million baby boomer retirements, 89% executive confirmation of talent shortage, and \$454 billion lost manufacturing value figures cited in Section 1.
4. IBM, cited in IIoT World, "IoT Data in Manufacturing: Fixing the 90% Waste Problem." Source of the estimate that approximately 90% of industrial data collected goes unused.
5. Zhang et al., "RAG-KG-IL: A Multi-Agent Hybrid Framework for Reducing Hallucinations and Enhancing LLM Reasoning through RAG and Incremental Knowledge Graph Learning Integration," arXiv:2503.13514, March 2025. Source of the ~73% and ~63% hallucination-reduction figures cited in Section 7.
6. McKinsey & Company, industrial reliability and maintenance value analysis, referenced in the value-driver ranges in Section 12.
7. The semantic/kinetic framing of operational ontologies in Section 4.6 — semantic elements (entities, properties, relationships) paired with kinetic elements (governed actions and decision capture) — reflects published architecture practice across the operational-ontology category.

Notes on figures

Performance ranges attributed to SymphonyAI deployments (downtime reduction, throughput improvement, ROI, asset life extension) reflect outcomes reported across a deployment portfolio and are not guarantees. Results at any site depend on baseline maturity, asset criticality, data availability, production economics and scope. Baselines should be established and agreed before implementation so that outcomes can be evidenced against them.

Figures and schematics in this paper are structural representations intended to convey architecture and information flow. They are not screenshots of the product interface.

