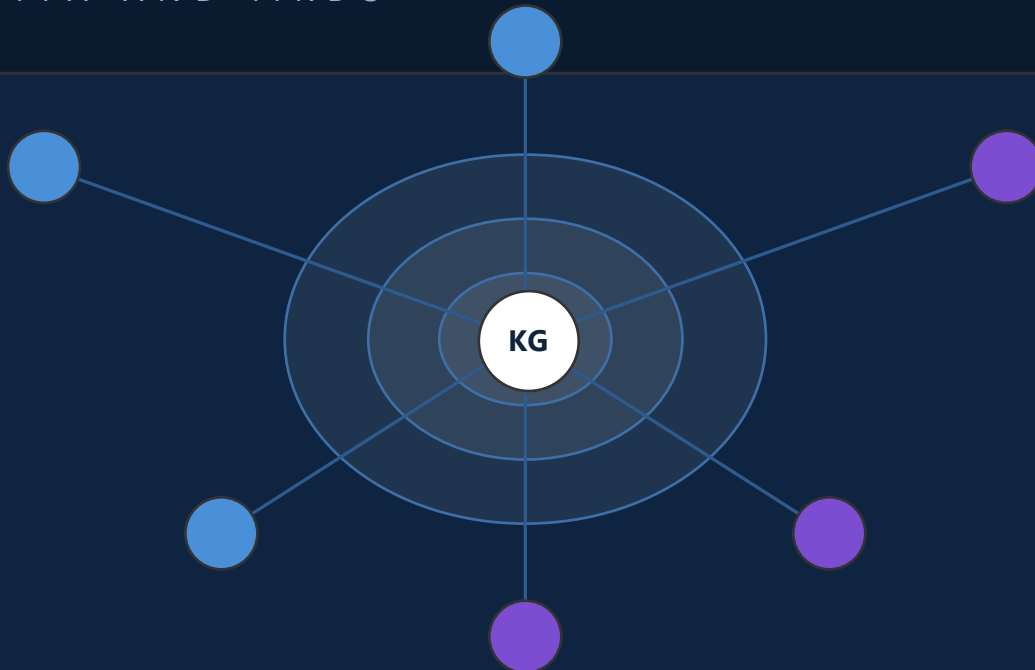


ADAPTIVE SEMANTICS

FOR DITA AND iRDS

WHITE PAPER



From Knowledge Graphs to Adaptive Semantics

Partly inspired by Saina Veigel's article "EvoOntology vs. iRDS: understanding two divergent ontology paradigms"

Evolving DITA and iRDS Digital Twins for AI

A plain-language introduction to how ontology evolution can complement knowledge graphs, inference engines, and AI agents without abandoning semantic governance.

CENTRAL IDEA

Inference extends what the knowledge graph knows. Ontology evolution extends how the knowledge graph can represent knowledge.

Contents

A guided tour from governed semantic models to an adaptive feedback loop in which AI-agent experience becomes evidence for improving the ontology itself.

01	The Foundation: Structured Semantics	Ontologies as a shared vocabulary for content, products and users
02	Ontology, Knowledge Graph, Language	Different roles, one semantic contract
03	From Ontologies to Digital Twins	Connecting graphs into a unified twin
04	What the Inference Engine Already Does	Deriving facts inside a known vocabulary
05	Where EvoOntology Changes the Model	Detecting the vocabulary
06	A simple Analogy	Routes, legends and scenic motorcycle roads
07	The Adaptive Semantic Feedback Loop	A second cycle around the governed core
08	Detecting Semantic Gaps	Seven signals from agent experience
09	Example: Operational Criticality	A new distinction, and what inference does with it
10	Governance is still essential	From candidate change to enterprise model
11	Building on GraphDB	Process innovation, not a new database
12	Adaptive Semantics in the Vector Index	Effective semantic views and incremental re-indexing
13	From Static to Adaptive Semantics	Designed, observed, continuously improved
14	Richer Semantics and more capable Agents	The eight-step improvement cycle
15	A new Role for the Digital Twin	Traditional versus adaptive
16	Strategic Implications	Five consequences for the enterprise
17	Conclusion	Adaptive semantics without abandoning governance

Introduction

Organizations are increasingly using artificial intelligence to help technicians, engineers, content developers, and other users find information, answer complex questions, and perform tasks. The effectiveness of those AI systems depends not only on the documents and data available to them, but also on how well that information is organized, connected, and understood.

DITA, iiRDS, and product or domain ontologies provide a strong semantic foundation for this environment. Together, they can describe technical content, information types, products, components, systems, procedures, users, metadata, and the relationships among them. When these structured models are connected through a knowledge graph, they create a digital representation of the technical information environment that AI agents can use for more intelligent retrieval and reasoning.

A traditional inference engine can extend this knowledge by deriving new facts from existing relationships and rules. However, it still operates within the semantic model that has already been defined. EvoOntology introduces an additional capability: recognizing when the semantic model itself is missing a concept, relationship, annotation, rule, or tool that an AI agent needs to perform effectively.

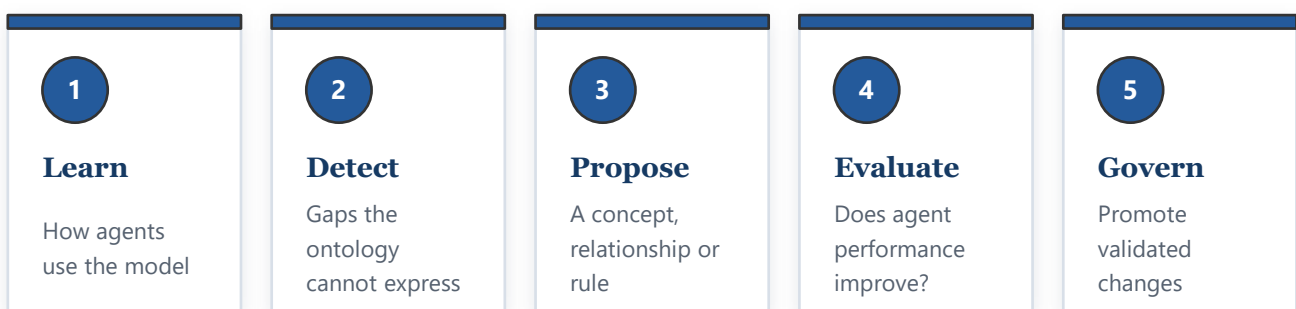
THE DISTINCTION

Inference extends what the knowledge graph knows.

Ontology evolution extends how the knowledge graph represents knowledge.

For a DITA and iiRDS digital twin architecture, EvoOntology is best viewed not as another independent ontology, but as an adaptive semantic feedback loop around the existing governed ontology and knowledge graph environment. It provides a mechanism for learning from actual AI agent usage, identifying gaps in the semantic model, proposing improvements, evaluating those improvements, and potentially promoting validated changes into the governed enterprise model.

The loop in five moves



One Overview: Governed Semantics, Digital Twins, and an Evolution Loop

Governed source ontologies feed digital twins and a unified knowledge graph. Inference and AI-agent activity generate usage evidence that can drive a governed ontology-evolution loop.

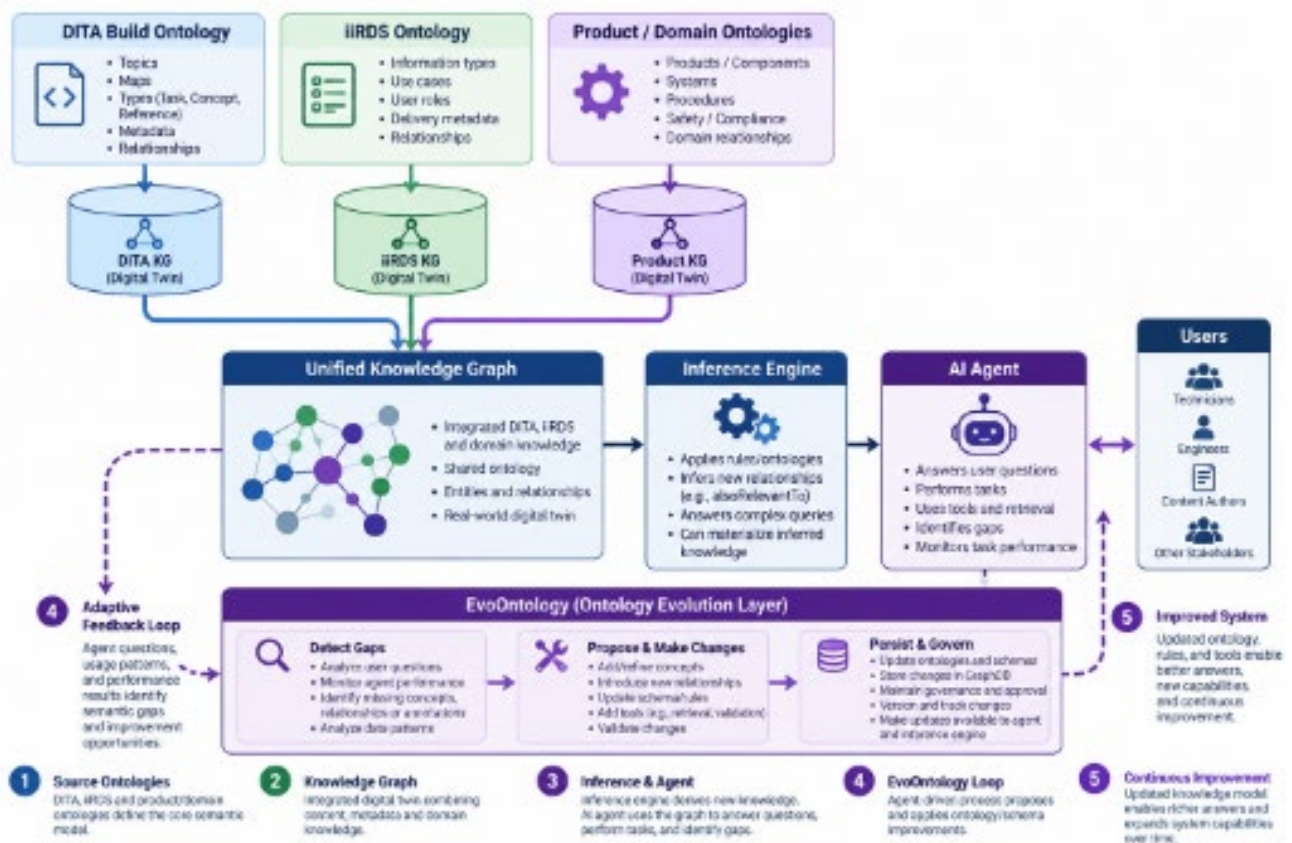


Figure 1. Adaptive semantic architecture for DITA and iIRDS digital twins. Governed source ontologies feed digital twins and a unified knowledge graph. Inference and AI-agent activity generate usage evidence that can drive a governed ontology-evolution loop.

01 The Foundation: Structured Semantics

An ontology can be thought of as a shared vocabulary for describing a particular world. It defines the important things in that world, the relationships among them, and in some cases the rules that determine how those concepts can be used.

In a technical information environment, several ontologies can work together. Each one describes a different slice of reality, and together they cover content, delivery context and the physical product world.

DITA ontology

Topics, maps, task and concept types, metadata, content relationships, and product applicability.

iiRDS ontology

Information types, use cases, user roles, delivery metadata, and contextual relationships.

Product / domain ontology

Products, systems, components, procedures, safety requirements, and engineering relationships.

LEADERSHIP PERSPECTIVE

The ontology is not simply a taxonomy. It provides a common language that allows content, products, metadata, AI, and business processes to understand one another.

What a shared vocabulary buys you

- Consistent meaning across authoring, delivery, engineering and AI systems
- Relationships that can be validated instead of guessed
- A contract that survives changes in tooling and programming languages
- A basis on which inference — and later, evolution — can operate

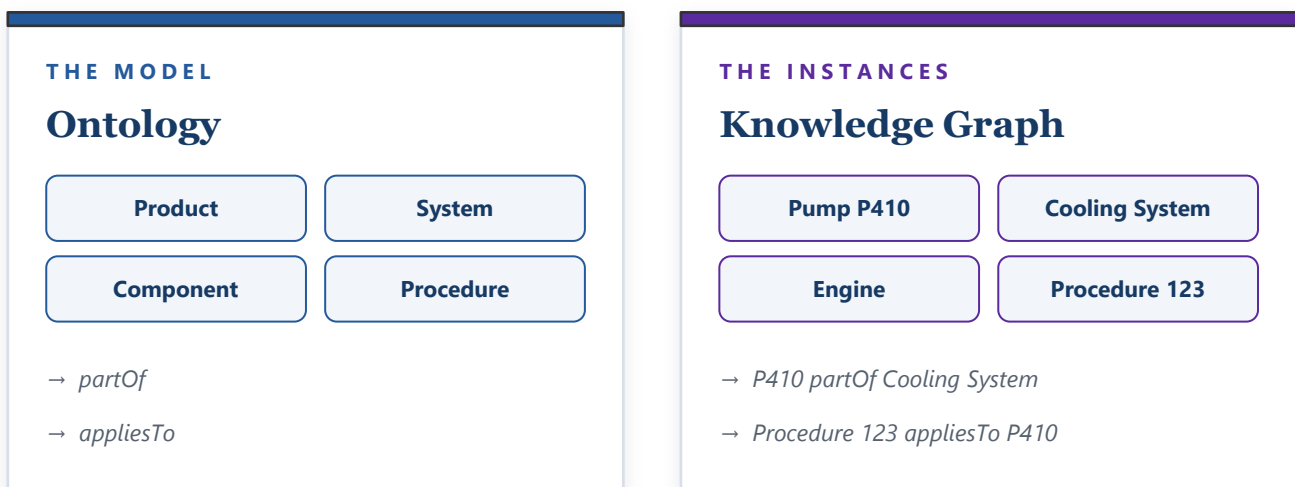
02 Ontology, Knowledge Graph, and Implementation Language

A knowledge graph can technically be created without a formal ontology. A graph can simply contain nodes and relationships. In this digital twin architecture, however, the ontology is what gives those nodes and relationships consistent meaning.

It defines the concepts the organization recognizes, the relationships that are valid, and the semantic rules that allow systems to interpret the graph in a predictable way.

■ Does the ontology generate the graph?

The ontology and the knowledge graph play different roles. The ontology defines the model of the world, while the knowledge graph contains the actual things in that world and the relationships among them.



The ontology does not need to create every graph statement by itself. Data pipelines, content processing, APIs, extraction tools, AI systems, and other applications can populate the graph. The ontology provides the semantic contract that tells those systems what the data means and how it should relate.

IN SHORT

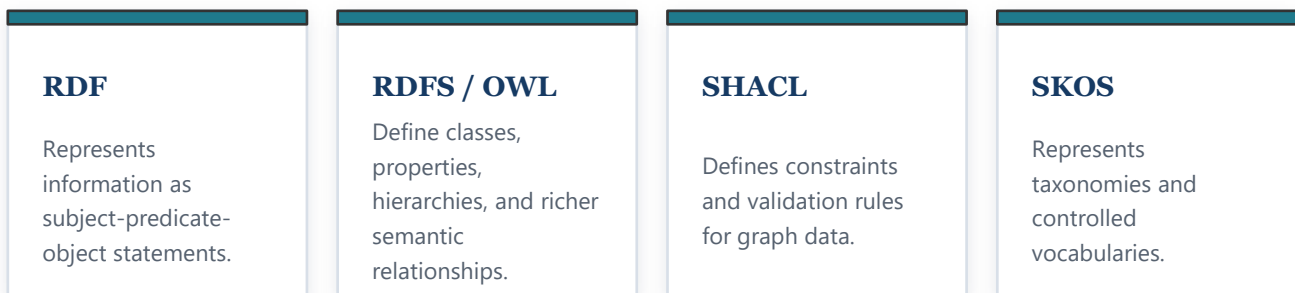
The model defines what may exist. The graph records what does exist. Neither replaces the other.

02 The Ontology is not Python-specific

The ontology is generally independent of the programming language used to implement the system. Python, Java, C#, JavaScript, and other languages are implementation technologies. They can create, query, update, validate, or reason over the same semantic model.

The ontology itself is typically represented using semantic standards such as RDF, RDFS, OWL, SHACL, or SKOS. These models can be serialized in formats such as Turtle, RDF/XML, or JSON-LD. A Python application and a Java application can therefore work with the same ontology and the same knowledge graph without requiring separate language-specific semantic models.

The standards toolbox



Why this matters for EvoOntology

An EvoOntology process might be implemented in Python, but the semantic change it proposes does not need to be Python-specific. For example, the process might identify that the model is missing a relationship such as `recommendedAfterReplacement` and propose adding that relationship to the shared ontology.

Once approved, that semantic change can be represented in the ontology and used by GraphDB, the inference engine, the vectorization pipeline, Java or Python applications, and AI agents. The shared ontology acts as the common semantic contract across all of those technologies.



Programming languages and libraries sit underneath these layers as implementation choices — which is why EvoOntology can improve the architecture without locking it to one software stack.

03 From Ontologies to Digital Twins

Each ontology can support a corresponding knowledge graph, or digital twin, containing real instances of the concepts defined by the ontology.

A DITA ontology can support a DITA knowledge graph, an iiRDS ontology can support an iiRDS knowledge graph, and a product ontology can support a product knowledge graph. These graphs can then be connected into a unified knowledge graph.



The unified graph connects technical information to the products, components, procedures, metadata, and user contexts that give that information meaning. This lets an AI system move beyond simple keyword matching and instead navigate relationships and context.



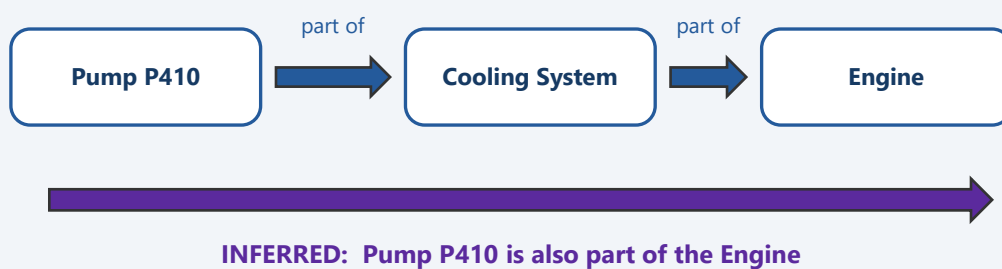
WHY IT MATTERS

Keyword search retrieves documents. A connected digital twin lets an agent follow relationships — from a user's situation to the component, to the procedure, to the applicable content.

04 What the Inference Engine already does

A knowledge graph does not need to contain every possible fact explicitly. An inference engine can derive additional knowledge from existing relationships and rules.

SIMPLE EXAMPLE



The same idea applies to technical content. If Procedure 123 applies to Pump P410, and Pump P410 is part of the Cooling System, a rule may allow the system to determine that Procedure 123 is also relevant to the Cooling System. That inferred relationship can be calculated when needed or materialized as a permanent relationship in the graph.

The limitation is important: the inference engine did not invent the concepts involved. It already knew what relationships such as applies to, part of, and also relevant to meant. It simply applied the ontology and rules that were already defined.

KEY DISTINCTION

An inference engine discovers additional knowledge using an existing vocabulary. It does not normally decide that the vocabulary itself needs to change.

What it can do

Derive facts, materialize relationships, apply rules and axioms, answer questions that were never stored explicitly.

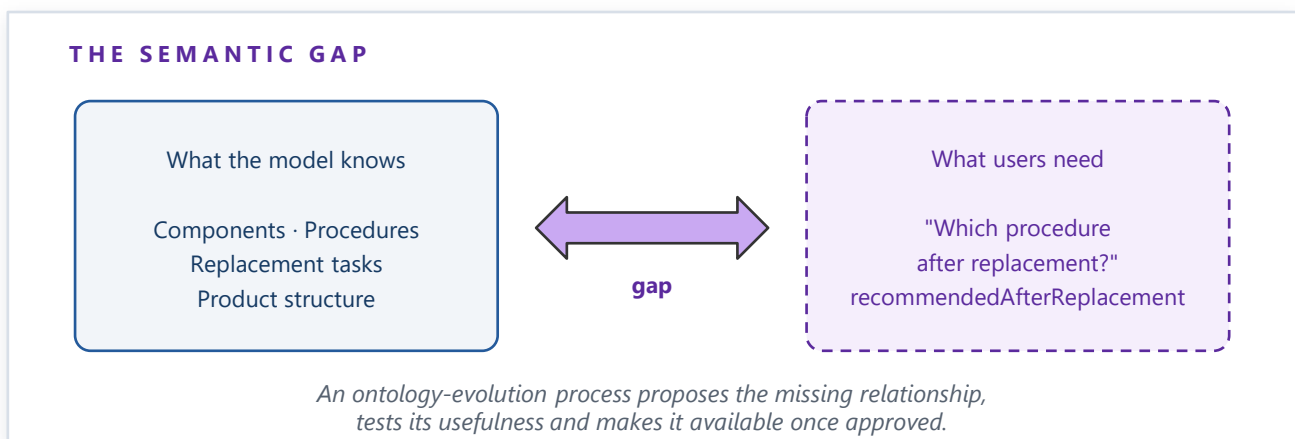
What it cannot do

Notice that a needed concept or relationship is absent from the model — and invent it.

05 Where EvoOntology changes the Model

EvoOntology operates at a different level. Instead of only asking what additional facts can be derived, an EvoOntology-style system can also ask whether the semantic model itself is missing something that the AI agent needs.

Consider a maintenance environment in which technicians repeatedly ask which procedure should be performed after replacing a component. The knowledge graph may already understand components, procedures, replacement tasks, product structures, and maintenance relationships. However, the ontology may have no relationship that expresses the idea recommended after replacement.



A conventional inference engine generally cannot decide to create a new property simply because users repeatedly need that relationship. An ontology-evolution process can identify this as a semantic gap, propose a relationship such as recommendedAfterReplacement, test its usefulness, and potentially make it available to the system.

THE CENTRAL IDEA

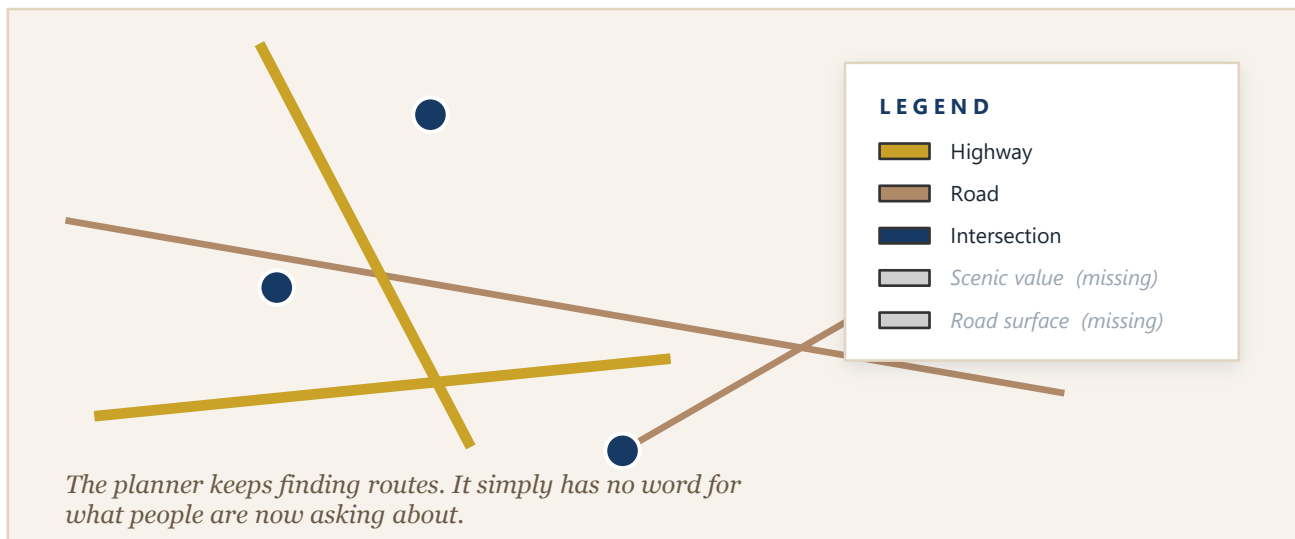
Inference extends what is known. EvoOntology can extend how knowledge is represented.



06 A Simple Analogy: the Map and its Legend

Think of the ontology as the legend and rules of a road map. It defines roads, highways, intersections, addresses, routes, and permitted turns. A route planner can calculate millions of routes that were never explicitly stored. That is inference.

Now imagine that users repeatedly ask for scenic motorcycle routes, but the map has no concept of scenic value, motorcycle suitability, road surface, or preferred riding conditions. The route planner can continue finding routes, but it does not have the vocabulary needed to represent what users are asking for.



MAP ANALOGY

Inference finds new routes using the existing map. Ontology evolution can improve the legend of the map.

07 The Adaptive Semantic Feedback Loop

The most significant architectural implication is that EvoOntology does not necessarily require a completely separate knowledge graph. The existing GraphDB-based architecture can remain the foundation.

■ Cycle one — the knowledge cycle



An EvoOntology-inspired architecture adds a second cycle in which agent experience becomes evidence about where the semantic model is weak or incomplete.

■ Cycle two — the evolution cycle



This creates a feedback loop between the AI agents that use the semantic model and the ontology that defines it. The agent becomes a source of operational evidence about what concepts, relationships, annotations, rules, or tools may be missing.

Same foundation

No separate knowledge graph is required; the governed environment remains the core.

New evidence source

Operational agent behaviour, not only modelling workshops.

Controlled promotion

Only validated, approved changes reach the enterprise model.

ARCHITECTURAL IMPLICATION

Adaptive semantics is a second cycle placed around the governed core — not a replacement for it.

08 Detecting Semantic Gaps through Agent Experience

AI agents provide something that traditional ontology-development processes often lack: continuous evidence about how the semantic model performs during real use.

Patterns in user questions and agent behaviour can reveal that the problem is not simply missing data, but missing semantic structure.

Seven signals worth watching

1

Incomplete answers

Questions that repeatedly produce incomplete answers

2

Unmodelled concepts

Concepts users frequently mention that are not represented

3

Rebuilt relationships

Relationships that must repeatedly be reconstructed

4

Query workarounds

Queries that require excessive workarounds

5

Missing classification

Missing annotations or classifications

6

Tooling gaps

Gaps in retrieval, validation, or other supporting tools

7

Weak spots

Areas where agent performance is consistently weak

READING THE EVIDENCE

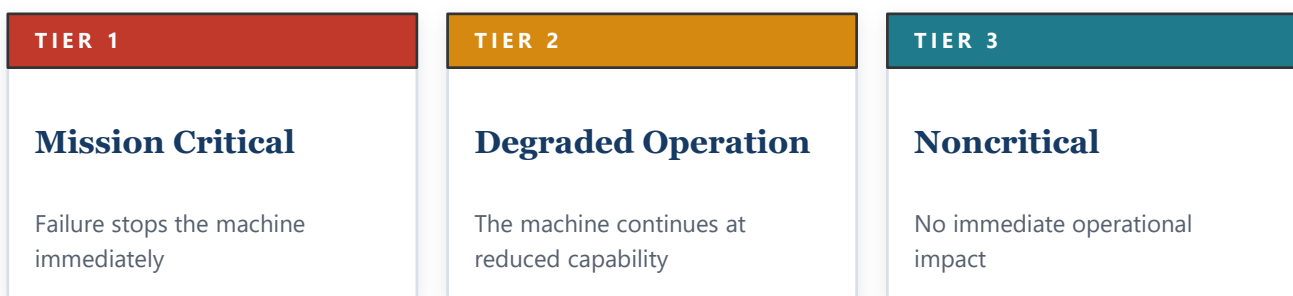
A missing answer is not always missing data. Recurring patterns often point to a vocabulary that cannot express what users are asking for.

09 Example: Introducing Operational Criticality

Suppose maintenance users frequently distinguish between components that can immediately stop a machine and components that allow the machine to continue operating in a degraded state. The product ontology may simply describe all of them as components, without representing operational importance.

An EvoOntology-style process could propose a concept called Operational Criticality, with classifications such as Mission Critical, Degraded Operation, and Noncritical. It could then introduce a relationship such as Component has Operational Criticality.

The proposed classification



Component has Operational Criticality → Tier



HOW THE TWO CAPABILITIES WORK TOGETHER

Ontology evolution introduces the new semantic distinction. Inference then uses that distinction to derive additional knowledge, such as whether a component failure requires immediate shutdown.

10 Governance is still essential

Adaptive semantics should not mean uncontrolled ontology changes. DITA, iiRDS, product, safety, regulatory, and engineering ontologies may represent carefully governed enterprise knowledge, and changes can affect many downstream systems.

■ The promotion path



This creates a practical balance. The governed semantic core remains controlled, stable, versioned, and trusted. An adaptive semantic layer can discover and test new concepts, relationships, rules, and tools without immediately changing the official model.

ADAPTIVE SEMANTIC LAYER

Discovers, proposes and tests new concepts, relationships, rules and tools — without changing the official model.



approved changes only

GOVERNED SEMANTIC CORE

Controlled, stable, versioned and trusted — DITA, iiRDS, product, safety and regulatory ontologies.

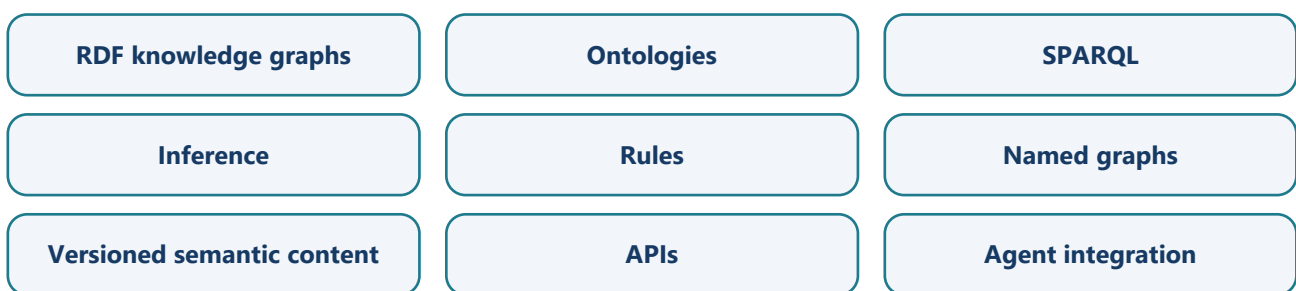
GOVERNANCE PRINCIPLE

AI can help discover what the ontology may be missing. Governance determines what becomes part of the enterprise ontology.

11 Building on the existing GraphDB Architecture

A major advantage of this approach is that it may not require replacing the existing knowledge graph technology. A GraphDB-based environment may already provide many of the necessary building blocks.

■ Building blocks already in place



If the architecture also permits controlled creation of classes, properties, axioms, rules, and other semantic structures, much of the EvoOntology approach could potentially be implemented within the existing environment. In that case, the key innovation is not a new database. It is a disciplined process for continuous ontology evolution.

■ The disciplined loop



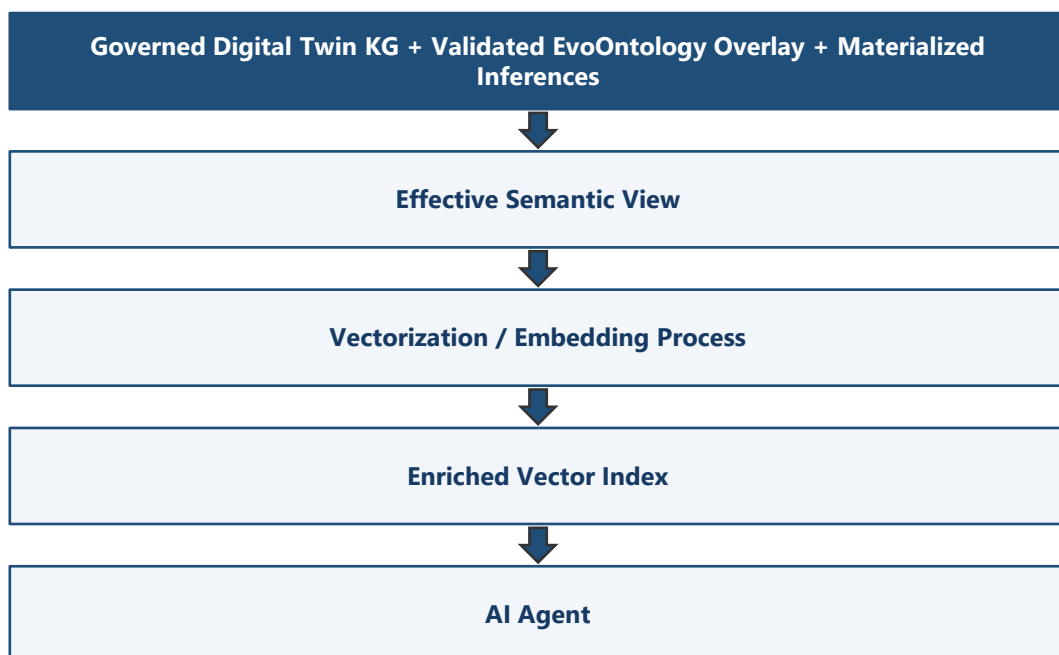
THE REAL INNOVATION

Not a new database — a disciplined, repeatable process for continuous ontology evolution inside the environment you already run.

12 Extending Adaptive Semantics into the Vector Index

The current architecture can continue to derive a vector index from the digital twin knowledge graph. EvoOntology does not replace that pipeline. Instead, it can provide additional validated semantic information that can be included in the source used to create embeddings.

The key design decision is which view of the graph is presented to the vectorization process. A practical pattern is to create an effective semantic view that combines the governed digital twin KG, validated EvoOntology semantic additions, and selected materialized inferences. That combined view can then be used to create an enriched vector index while the governed core remains stable.



DESIGN DECISION

Which view of the graph is presented to the vectorization process determines what the embeddings can express.

12 What EvoOntology Intelligence can carry into the Vector Index

The vector index will contain only the information that the embedding pipeline receives. If approved EvoOntology changes are included in the effective semantic view, new concepts, relationships, annotations, entity facts, and materialized inferred facts can all contribute to richer embeddings and better semantic retrieval.

EvoOntology addition	Vector index benefit	Primary execution layer
New concepts or classes	Yes	Vector retrieval + KG
New relationships	Yes	Vector retrieval + KG
New entity facts	Yes	Vector retrieval + KG
Materialized inferred facts	Yes	Vector retrieval + KG
Annotations and descriptions	Strongly	Vector retrieval
Rules, axioms, and constraints	Only partially	Inference engine / reasoner
New tools or runtime behavior	Not directly	AI agent / tool layer

Rules, axioms and constraints stay with the reasoner; new tools and runtime behaviour belong to the agent layer. The vector index benefits most from new concepts, relationships, facts and rich annotations.

12 Why the Vector Index does not replace Reasoning

Embeddings are excellent for finding information that is semantically similar to a user question, but they are an approximate representation of meaning. They do not preserve every formal relationship, constraint, or logical rule with the precision of the knowledge graph.

For that reason, vector retrieval and symbolic reasoning should remain complementary.

VECTOR RETRIEVAL ANSWERS

What information is semantically relevant to this question?

KNOWLEDGE GRAPH REASONING ANSWERS

What relationships and facts can be logically established?

EVOONTOLOGY ANSWERS

Is the current semantic model sufficient, and if not, how should it improve?

Incremental re-indexing after semantic evolution

Approved semantic changes do not necessarily require rebuilding the entire vector index. A practical implementation can version semantic changes, identify the affected entities, relationships, topics, or graph neighborhoods, and re-embed only the material influenced by the change. This keeps the vector index synchronized with the evolving semantic model while limiting processing cost.



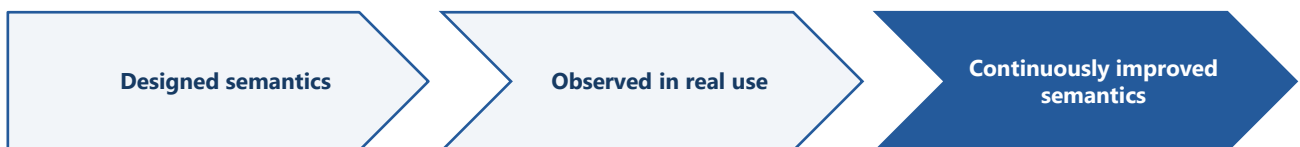
KEY POINT

EvoOntology replaces neither the digital twin KG, the inference engine, nor the vector index. It adds an adaptive semantic layer that improves all three.

13 From Static Semantics to Adaptive Semantics

Traditional enterprise ontologies are designed by people who attempt to anticipate the concepts and relationships an organization will need. That work remains essential. However, no design team can anticipate every question that future AI agents and users will ask.

An adaptive architecture provides another source of insight. Actual AI usage reveals where the model works well and where it does not. The architecture can therefore move from designed semantics to designed, observed, and continuously improved semantics.



IMPORTANT POINT

Adaptive semantics does not replace ontology engineering. It adds a feedback mechanism to ontology engineering.

14 Richer Semantics and more capable Agents

AI agents are increasingly expected to do more than search for documents. They may answer complex technical questions, troubleshoot equipment, assemble procedures, identify applicable content, understand product configurations, determine safety implications, recommend next steps, retrieve supporting evidence, and interact with enterprise tools.

The richer the semantic model, the more effectively the AI agent can understand the context surrounding those tasks. The knowledge graph provides structured knowledge. The inference engine expands that knowledge. The adaptive ontology layer identifies where the structure itself needs to improve.

■ The eight-step improvement cycle

- 1 Users interact with the AI agent.
- 2 The agent uses the knowledge graph and inference engine.
- 3 The agent encounters semantic limitations.
- 4 The adaptive layer identifies and analyzes those limitations.
- 5 New concepts, relationships, rules, annotations, or tools are proposed.
- 6 Changes are evaluated and governed.
- 7 Approved improvements become available to the knowledge graph and AI agents.
- 8 Agent performance improves, while new usage creates additional learning opportunities.

15 A new Role for the Digital Twin

A traditional digital twin represents a product, system, process, or information environment. An adaptive semantic digital twin can go further by gradually improving the language it uses to describe that environment.

Traditional Semantic Digital Twin		Adaptive Semantic Digital Twin
Represents known concepts	⇒	Can identify missing concepts
Uses defined relationships	⇒	Can propose new relationships
Uses established rules	⇒	Can identify where new rules are needed
Supports inference	⇒	Supports inference plus semantic evolution
Changes through formal modeling	⇒	Can learn from agent usage
Primarily design-driven	⇒	Design-driven and experience-informed
Knowledge grows	⇒	Knowledge and semantic structure can grow

THE SHIFT

The twin no longer only represents the world. It also improves the language it uses to describe that world.

16 Strategic Implications

Five consequences for organizations that already invested in structured content, ontologies and knowledge graphs.

01**Existing semantic investments become more valuable**

DITA, iIRDS, product ontologies, and knowledge graphs remain the foundation on which adaptive AI capabilities can be built.

02**AI becomes a consumer and a source of semantic insight**

Agents consume structured knowledge, but their performance also reveals where the semantic model needs improvement.

03**Ontology development can become more evidence-based**

Ontology teams can use patterns in real user questions and agent behavior as evidence for future modeling decisions.

04**Governance remains central**

Adaptive discovery can accelerate innovation while formal approval processes protect enterprise consistency and trust.

05**The semantic model can evolve with the business**

New products, tasks, regulations, and operating conditions can create semantic requirements that were not anticipated when the ontology was first designed.

17 Conclusion

The ontology also serves as a language-independent semantic contract. It defines what the concepts and relationships mean, while Python, Java, GraphDB, embedding models, and other technologies implement or consume that shared model.

This separation allows the semantic architecture to evolve without tying the enterprise knowledge model to a particular programming language or application stack.

What the architecture already provides

DITA Defines structured technical content.	iiRDS Adds information-delivery and contextual semantics.	Product ontologies Describe the physical and operational environment.
Knowledge graphs Connect those worlds.	Inference engines Derive additional knowledge.	AI agents Answer questions and perform tasks.

A vector index derived from this governed and adaptively enriched semantic view can continue to support fast semantic retrieval for AI agents. EvoOntology introduces one additional capability: a feedback loop through which the system can identify when its existing semantic model is no longer sufficient.

The architecture can therefore move from a one-directional flow of ontology to knowledge graph to inference into a continuous improvement cycle that incorporates AI-agent experience.

THE TAKEAWAY

Inference extends what the graph knows. Ontology evolution extends how the graph can represent knowledge.

A practical Path to Adaptive Semantics

For DITA and iiRDS digital twins, this suggests a practical path toward adaptive semantics without abandoning governance.

Enterprise ontologies

Remain the trusted, governed foundation.

Knowledge graph

Remains the connected representation of enterprise knowledge.

Inference engine

Remains responsible for deriving new facts.

AI agent

Becomes both a consumer of semantic knowledge and a source of evidence about where the model can improve.

THE RESULT

Not simply a larger knowledge graph, but a knowledge architecture that can become progressively better at representing the concepts and relationships required by the people, processes, and AI systems that depend on it.

ADAPTIVE SEMANTICS

Inference extends what the graph knows. Ontology evolution extends how the graph can represent knowledge.