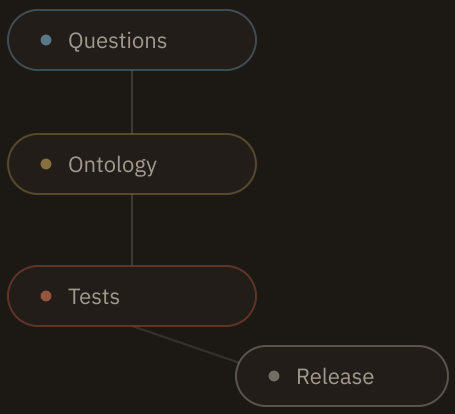
Palmetto Interactive

WHITEPAPER · SEMANTIC DEVELOPMENT

Meaning, under version control



Every AI team is already writing definitions. They live in prompts, tool descriptions, and the heads of a few senior people. Semantic development is the discipline of writing them down once, testing them, and shipping them like code.

Matt Lucas · Founder & Principal, Palmetto Interactive
Twenty years across enterprise risk, finance and compliance technology, and enterprise knowledge-graph delivery.

OCTOBER 2026

01 · THE PROBLEM

You’re already doing this, just badly

Open any production agent and read its system prompt. You’ll find a glossary. It says what an active customer is, which accounts count, and what "exposure" means here.

Those definitions are semantic development. They’re just done in the least governed place available. Nobody versions them. Nobody tests them. And when a second team builds a second agent, it writes its own glossary, slightly different from the first.

Tool descriptions are the same story. Every function an agent can call carries a little schema and a paragraph of business rules. Multiply that by fifty tools and a dozen agents, and the business’s meaning is scattered across hundreds of strings no one owns.

This paper is about moving that meaning somewhere better. One place, with owners, tests, and releases. Our companion paper on knowledge graphs covers the engines and the retrieval side. This one covers how the model gets built and kept right.

02 · THE COUNTER-ARGUMENT, TAKEN SERIOUSLY

When a metrics layer is enough

Plenty of organizations don’t need an ontology. They need consistent numbers. And the metrics-layer products got much better in the last year.

Snowflake semantic views became queryable with standard SQL in March 2026. Databricks registers metric views in Unity Catalog. dbt Labs open-sourced MetricFlow in 2025.[1] If your agents mostly ask "what was revenue by region last quarter," that’s the right tool, and the cheaper one.

There’s also a shared spec coming. The Open Semantic Interchange effort, now Apache Ossie, wants metric definitions to move cleanly between platforms. It’s still incubating at Apache, with a release candidate and no final release.[2] Worth watching. Not yet worth betting a program on.

WHERE A METRICS LAYER WINS

- Questions are calculations over well-modeled tables
- One warehouse, one set of keys
- The goal is "one definition of revenue" across BI and agents
- Nobody needs the system to reason about types or relationships

WHERE IT RUNS OUT

- The same customer, asset, or contract lives in five systems
- Meaning depends on type. A guarantor is a party, but not every party can guarantee
- Agents need to *act*, and actions need rules
- Definitions must line up with an industry standard or a regulator

Most enterprises end up with both. A metrics layer for calculations, an ontology for what things are and how they connect. The work is keeping the two pointed at the same definitions.

03 · REQUIREMENTS

Start with the questions

Every serious ontology method agrees on one habit. Before you model anything, write down the questions the model has to answer. The field calls them competency questions. The idea goes back to Grüninger and Fox in 1995, and it has outlived every method built on top of it.[3]

A good competency question is specific and checkable. "Which controls mitigate risks rated high in a business unit with an open incident?" You can turn that into a query, run it, and see whether the model answers it. That’s a test.

The named methods mostly differ in how much process they wrap around that habit. We take what’s useful from each and skip the paperwork.

METHOD	WHAT WE TAKE FROM IT
METHONTOLOGY (1997)	A full lifecycle with documented stages. Useful when a public-sector or standards body needs to see the process. Heavy for anyone else.
NeOn	Scenarios built around reuse. Most enterprise work is aligning what already exists, and NeOn is honest about that.
LOT (2022)	A lightweight, industry-oriented workflow. Requirements, implementation, publication, maintenance. The closest fit to how product teams already ship.
eXtreme Design	Small, reusable design patterns and test-first iterations. Keeps a model reviewable when several people work on it.
Test-driven ontology development	Competency questions and constraints written as executable tests. This is the piece that matters most in 2026, because it’s what lets you change the model safely.

One honest note. We found no good data on how widely any of these named methods is used in industry today. We recommend the habits because they work in our delivery, not because a survey says so.[4]

04 · REUSE

Don’t start from a blank page

The most expensive ontology is the one you invent from scratch. Somebody has already argued about what a "party" or an "event" is, and most of those arguments are written down.

STARTING POINT	WHAT IT IS, AND WHEN IT FITS
gist	A small business upper ontology from Semantic Arts. Around 100 classes, plain language, CC BY 4.0. Version 14.1 shipped in April 2026. A sensible default for an enterprise with no industry standard to follow.[5]
FIBO	The financial industry ontology, hosted by the EDM Council and standardized through OMG. Quarterly releases, the latest in October 2026. Large. Teams take the modules they need and leave the rest.[6]
IOF Core	The Industrial Ontologies Foundry's mid-level ontology for manufacturing and supply chain. Built on BFO.
BFO and CCO	A formal top-level ontology and the mid-level suite built on it. Common in defense, biomedical, and research consortia. Worth it when you have to interoperate with that ecosystem. Costly without formal-ontology skills on the team.[7]
schema.org	The web's shared vocabulary, version 30.1 as of September 2026. A good seed for entity types and public-facing data. Not an enterprise upper ontology.

Reuse has a cost too. Every term you import is a term your people have to understand. So we import narrowly, document why each import exists, and pin the version.

05 · NAMING THE DIFFERENCE

Four layers that get mixed up

A lot of failed semantic programs tried to make one artifact do four jobs. Separating them is most of the design.

LAYER	WHAT IT HOLDS	TYPICAL STANDARD OR TOOL
Vocabulary	Agreed terms, synonyms, and broader-narrower hierarchies	SKOS, a W3C Recommendation since 2009
Ontology	What kinds of things exist, how they relate, and the rules that always hold	OWL 2, or a typed schema in LinkML or TypeDB's TypeQL
Shapes	What valid data looks like in a given system or pipeline	SHACL. The 2017 Recommendation is the standard today. Version 1.2 is still in Working Draft.[8]
Metrics	Governed calculations over the data	MetricFlow, Snowflake semantic views, Databricks metric views

The split that matters most is ontology versus shapes. The ontology says what's true about the world. A shape says what this particular dataset must contain before you trust it. Mixing them is how teams end up with a "customer" class that fails half the time, because one system doesn't capture email.

06 · ENGINEERING PRACTICE

Treat the ontology like code

This is the biggest change in how good teams work. The ontology lives in Git. Every change is a pull request. CI runs the tests. Releases get version numbers, and consumers pin to one.

The tooling for this is mature. The biomedical world has run ontologies this way for years, using the Ontology Development Kit and ROBOT to build, check, and release in a pinned container. ROBOT shipped version 1.9.11 this October.[9]

LinkML has become the practical bridge for teams that think in data models rather than logic. You write one schema in YAML and generate JSON Schema, SHACL, Python classes, and OWL from it. It's on version 1.12, with a peer-reviewed paper published in January 2026 and real uptake across biomedical research programs.[10]

TypeDB takes a different route. It's a database with a strongly typed schema, checked when data is written. Its TypeQL language supports inheritance, cardinality and value constraints, and reusable query functions. Version 3.0 was a full rewrite in Rust, and the 3.x line ships often.[11] If your team wants the database itself to refuse bad data, that's an appealing model. You give up the open-world reasoning and portability of OWL.

WHAT A RELEASE GATE CHECKS

Every competency question still returns the expected answer. Every shape still validates the reference data. No class lost its definition or owner. Nothing deprecated got deleted outright. And the change log says, in plain words, what moved and who approved it.

07 · LLMS IN THE LOOP

What the research says models can do

Language models are useful ontology assistants. The evidence is just as clear about where they stop being useful.

A useful 2025 result comes from Lippolis and colleagues. They gave models competency questions and asked for ontologies. A frontier model with a structured prompting method beat novice ontology engineers on their benchmark. But the authors also logged recurring errors. Wrong domains and ranges, broken inverse relationships, cycles in the class hierarchy, and plenty of superfluous classes.[12]

A September 2026 study tested nine models across zero-shot, iterative, and multi-agent setups. Its headline finding is the clearest summary we've read. Models recover vocabulary terms more reliably than they build ontologies. The gap was widest on properties, hierarchy, and axioms.[13]

The LLMs4OL challenge at ISWC 2025 points the same way. Some tasks scored above 0.9 F1. Others scored near zero. Relation extraction was the hardest task in the first round. In the second, hybrid pipelines did better than any single model.[14]

Matching ontologies to each other shows a useful pattern. One well-placed 2025 system used a language model only to check the mappings the matcher was least sure about. It ranked in the top two on a biomedical track of the OAEI evaluation.[15]

> Use the model to draft and to check. Keep people on structure and rules.

So here's how we use them. Models draft vocabulary from documents and propose competency questions. They extract entities against a fixed schema. They review uncertain mappings. People own the class hierarchy, the relationships, and the rules, and every model-drafted change goes through the same pull request as everything else.

Read the fine print on "ontology grounding"

Several memory and extraction tools now take an ontology as input. We use cognee ourselves, as shared memory for our own agents, and it's good. Its ontology support also reads only part of OWL. Classes, subclasses, types, and object properties count. Datatype properties are ignored, and its default mode keeps entities that match nothing.[16] Neo4j's LLM Graph Builder accepts node and relationship labels, and it's a Labs project rather than a supported product. That's a reasonable place for these tools to be. Just don't mistake label hints for validation. If you need data checked against the model, run SHACL or a typed schema on the output.

08 · ONTOLOGIES AGENTS ACT THROUGH

From meaning to action

The newest idea in this space is also the most practical. An ontology can describe what an agent is allowed to *do*, as well as what things mean.

Palantir built its platform around this. Its ontology has object types and link types, plus action types that change data under governance and functions that hold business logic. Its agent tooling calls those actions directly, with optional user confirmation.[17] Microsoft's Fabric IQ ontology, in preview this fall, takes a similar shape. Entity types, relationships, metrics, rules, and bindings to live data, with agent changes proposed before they're applied.[18]

For anyone accountable for agents, this is the useful framing. An action type is a permission boundary with a definition attached. "Approve exception" means one specific, logged change to one type of object, with rules about who can trigger it. The model can't improvise a new action, because the ontology doesn't have one.

You don't need either platform to work this way. The pattern carries over. Name the actions in the model, type their inputs, attach the rules, and expose only those actions as agent tools.

09 · GETTING DATA IN

Map before you migrate

The usual first instinct is to copy everything into a new graph. Often you don't need to.

Virtual knowledge graphs leave data where it is. You write mappings from source tables to the ontology, and queries get rewritten to SQL against the original database. R2RML has been the W3C standard for those mappings since 2012. Ontop is the mature open-source engine for it, on version 5.5. RML extends the idea to JSON and CSV, as a community draft rather than a W3C standard.[19]

Starting virtual lets you prove the model against real data in weeks, without a new copy to secure and keep in sync. You materialize into a graph store later, for the parts that need fast traversal, history, or entity resolution across sources.

10 · GOVERNANCE

Who gets to change a definition

The technical side of change control is solved. Version numbers, deprecation flags, release notes. The people side is where programs stall.

Every class and relationship needs a named business owner, someone who can settle an argument about what it means. Every change needs a reviewer on the technical side, who checks it won’t break consumers. And deprecated terms stay in the model, marked and pointing at their replacement, for at least one full release.

The reference ontologies show what a steady rhythm looks like. FIBO releases quarterly. gist ships roughly one major version a year, with a strict rule about what goes in its namespace. Pick a cadence your business owners can actually keep, and publish it.

11 · WHERE YOU ACTUALLY ARE

A maturity check worth scoring honestly

STAGE	WHAT IT LOOKS LIKE FROM THE INSIDE
1 · Implicit	Definitions live in prompts, tool descriptions, report logic, and people's heads. Each agent team writes its own.
2 · Written	A glossary or catalog exists. It's documented, not enforced, and nothing tests it.
3 · Modeled	One domain has an ontology in Git, built from competency questions, reusing a reference model. Shapes validate one pipeline.
4 · Released	Versioned releases gated by tests. Named owners. Agents and the metrics layer read from the same definitions.
5 · Operational	Agent actions are defined in the model. New domains onboard through a known process in weeks.



Most teams we meet are at one or two, and the right next step is three on a single domain. You can reach it in one engagement. And because it ships with tests, the second domain costs a lot less than the first.

12 · A WORKED EXAMPLE

Risk, controls, and incidents

Operational risk assessment at a top-10 North American bank spanned disconnected taxonomies, controls, and incident data across the enterprise. Assessment and reporting needed manual aggregation to pull them together.

Engagement led by Palmetto's principal in a prior role. Presented as professional experience, not as a Palmetto client result or endorsement.

The engagement connected risks, controls, processes, and incidents into one structure people could assess. What came out was a connected operational-risk view supporting assessment and reporting that used to be assembled by hand.

The lesson that carries over is about the taxonomies. Each one was reasonable on its own. The work was agreeing how they related, and writing that agreement down where software could enforce it. That's semantic development in one sentence, and it's the part no tool does for you.

13 · WHERE TO START

Twenty questions and a repository

Pick the domain where inconsistent definitions cost you the most today. Write twenty competency questions with the people who own the answers. Choose a reference ontology to start from. Put the model in Git with the questions as tests. Then point one agent at it and retire the glossary in its prompt.

If writing those twenty questions is hard, that's the finding. It usually means nobody is empowered to settle what the terms mean, and you'd rather learn that in week two than month six.

How Palmetto works on this

We start with the decisions that matter and the questions your model has to answer, then model only as far as those uses justify. We reuse standards where they fit and say so when they don't. Every engagement opens with a bounded diagnostic ending in a decision brief.

The ontology, its tests, the release pipeline, the decision log, and the governance playbook all go to your team. If a metrics layer is the honest answer for you, the brief says that, and we can help you build that instead.

Palmetto Interactive palmettointeractive.com · support@palmettointeractive.com

REFERENCES & NOTES

1. Snowflake release notes: semantic views GA June 2025, standard SQL querying GA 2 March 2026. Databricks documentation, metric views in Unity Catalog. dbt Labs, "The OSI spec updates," 29 January 2026, on open-sourcing MetricFlow. Vendor sources.

2. Apache Ossie (Incubating), "Ossie enters the Apache Incubator," 10 July 2026. The project repository carries one release-candidate tag and no final release as of October 2026. Incubation isn’t endorsement by the ASF.
3. M. Grüninger and M. S. Fox, competency questions, IJCAI’95 Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.
4. Fernández, Gómez-Pérez and Juristo, METHONTOLOGY, AAAI Spring Symposium SS-97-06, 1997. Suárez-Figueroa, Gómez-Pérez and Fernández-López, the NeOn Methodology. Poveda-Villalón et al., "LOT: An industrial oriented ontology engineering framework," Engineering Applications of AI 111, 2022. Presutti et al., eXtreme Design. Keet and Ławrynowicz, "Test-Driven Development of Ontologies," ESWC 2016.
5. Semantic Arts, gist v14.1.0, released 17 April 2026, github.com/semanticarts/gist.
6. EDM Council, FIBO release 2026Q3, published 20 October 2026, github.com/edmcouncil/fibo.
7. BFO-2020 (ISO/IEC 21838-2), github.com/BFO-ontology/BFO-2020. Common Core Ontologies v2.2, August 2026.
8. W3C Data Shapes Working Group publications. SHACL 1.2 Core and related documents are Working Drafts as of October 2026. SHACL (2017) remains the Recommendation.
9. INCATools Ontology Development Kit and ROBOT, release v1.9.11, 1 October 2026.
10. LinkML v1.12.1, October 2026. "LinkML: an open data modeling framework," GigaScience, January 2026, doi:10.1093/gigascience/giaf152. Adopters as listed by the project.
11. TypeDB 3.0 released December 2024. Release 3.14.1, 22 October 2026, github.com/typedb/typedb.
12. Lippolis et al., "Ontology Generation using Large Language Models," arXiv:2503.05388, March 2025. Results on a 10-ontology, 100-question benchmark.
13. Li, Wang, Garijo and Poveda-Villalón, "CQ4OE," arXiv:2609.26029, September 2026.
14. LLMs4OL 2025 challenge overview, ISWC 2025, doi:10.52825/ocp.v6i.2913. Benchmark data derived from existing ontologies, not enterprise corpora. Mean scores mix subtasks with uneven coverage, so don’t read any single figure as "LLM accuracy."
15. Lushnei et al., "Large Language Models as Oracles for Ontology Alignment," arXiv:2508.08500. OAEI 2025 Bio-ML track. OAEI has no dedicated LLM track.
16. cognee documentation, "Ontology support." Neo4j Labs, LLM Knowledge Graph Builder README, which describes it as experimental and outside Neo4j’s supported products.
17. Palantir Foundry documentation, Ontology overview and Agent Studio tools. Vendor source.
18. Microsoft Learn, Fabric IQ ontology overview, updated 16 October 2026. Preview feature. Vendor source.
19. W3C R2RML Recommendation, 27 September 2012. Ontop 5.5.0, February 2026. RML-Core Draft Community Group Report, March 2026, which states it isn’t on the W3C standards track.

Version numbers and release dates were checked against project repositories and documentation on 30 October 2026, and they move quickly. Vendor documentation describes vendor products and is marked as such. Where a standard is still in draft, we say so rather than round it up.