Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

Competency Questions: From Business Need to Ontology Test

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-09-22 · competency questions ontology · ontology engineering · ontology requirements · sparql · shacl · knowledge graphs · requirements traceability · ontology testing

Original article: https://reliablecontext.com/articles/competency-questions-ontology-traceability-contract



In this article

1. Executive Summary
2. Introduction and Background
3. Definitions and the five-question taxonomy
4. From a business need to a precise competency question
5. The traceability contract
6. Verification forms, fixtures, and expected answers
7. Fixtures and worked examples
8. Regression, change, and retirement
9. Data Analysis and Evidence
10. Implications and Future Directions
11. Frequently Asked Questions (FAQs)
12. Conclusion

Executive Summary

An ontology competency question (CQ) is a natural-language question used to state what knowledge an ontology should cover or what an implemented ontology should be able to answer. The idea entered the ontology-engineering literature as a way to define scope, and Stanford's practical guidance still describes early CQs as sketches that need not be exhaustive (eil.mie.utoronto.ca) (protege.stanford.edu). That history matters because a useful CQ is not necessarily an executable query. QuO distinguishes **five types**, Scoping, Validating, Foundational, Relationship, and Metaproperty, with different purposes (arxiv.org. A valuable scope question and an executable validation question may therefore need different acceptance evidence.

The recommended unit of work is therefore a **traceability contract**, not a question string. It records the decision and accountable owner, CQ text and type, bounded scope, permitted evidence, reasoning regime, fixture and version, formal verification form, expected answer, counterexample, status, and review trigger. The business decision comes first. A question with no action, consequence, or owner is a topic prompt, not yet an acceptance requirement. The method works before an ontology exists because the decision, question type, scope, answer form, and evidence can already be agreed. It remains useful after implementation because the same record links requirement, test fixture, query or review, result, and change history. NeOn guidance similarly grouped CQs with answers and priorities, and linked them to derived concepts for traceability (research-archive.stem.open.ac.uk).

The quantitative evidence argues against assuming one question equals one query. A published dataset contains **234 CQs** but **131 SPARQL-OWL formalizations** (pubs.cs.uct.ac.za). The nine most frequent query patterns covered **63.1%** of its query set, but the mapping from question wording to formal patterns was not one-to-one (arxiv.org). A finite fixture can show that one declared model, dataset, query, and reasoning configuration produced the expected result. It cannot prove that the ontology is complete for the world.

Execution must match the claim. **SELECT** compares bindings, **ASK** checks whether a solution exists, an entailment test checks consequences under a named reasoning regime, and **SHACL** checks a data graph against a shapes graph (www.w3.org). Under the OWL open-world assumption, absence can mean unknown

rather than false (www.w3.org). This is why every contract needs explicit unknown handling, negative and boundary fixtures, and a review rule that distinguishes a changed requirement from a regression.

Introduction and Background

Competency questions sit at an unusually important boundary. On one side are stakeholder needs stated in ordinary language. On the other are ontology terms, axioms, data, queries, reasoners, validation rules, and test reports. Treating a CQ as only a sentence loses the implementation evidence. Treating it as only a SPARQL query loses the decision that made the query worth maintaining. The aim is durable acceptance evidence, not merely a successful demonstration query.

The original enterprise-modeling method introduced CQs as a basis for defining ontology scope (eil.mie.utoronto.ca). Later practice preserved that intentionally preliminary role. Ontology Development 101 says the initial questions need not be exhaustive and that application needs determine model granularity (protege.stanford.edu). A requirements-engineering study likewise describes ontology scoping alongside process models, scenarios, software requirements, and domain-expert consultation, rather than as a query-writing exercise alone (link.springer.com). The Linked Open Terms methodology starts from documentation and data provided by domain experts and users, then assigns them a role in validating requirement correctness and completeness (lot.linkeddata.es).

This report operationalizes that boundary as a **traceability contract**. The term is a proposed practitioner construct, not a claim that one universal standard already exists. The contract is durable if a reviewer can follow the chain backward from a result to the query, fixture, reasoning assumptions, question, source need, owner, and decision, and forward again after any part changes. NASA's general requirements guidance offers a useful analogue: requirements should have unique identifiers and definitive sources, while bidirectional traceability helps identify affected design, documentation, code, and tests (www.nasa.gov) (swehb.nasa.gov).

Definitions and the five-question taxonomy

An **ontology competency** is the capability expected of an ontology in a defined use context. A **competency question** expresses some part of that expectation as a natural-language question. A **fixture** is a controlled ontology and data state used to exercise the expectation. A **reasoning regime** states what consequences may be inferred beyond explicit triples. An **expected result** defines the observable condition that counts as acceptance.

These definitions prevent a recurrent category error. The CQ is the requirement-facing question. A SPARQL query, Shapes Constraint Language (SHACL) shape, entailment assertion, or expert-review protocol is a verification artifact. The same CQ may have more than one verification artifact, and some useful CQs should not be forced into query form. QuO's five-part classification makes this explicit. Its initial Repository of Ontology Competency QuestionS (ROCQS) contains **438 questions** across the types (arxiv.org).

Table 1 classifies a proposed question before anyone chooses a query language.

CQ type	Purpose	What acceptance can mean	Common mistake
Scoping	Establish which domain entities or concerns belong inside the ontology's intended coverage. QuO says this type contributes to describing ontology scope (arxiv.org).	Sponsor and domain-expert agreement, a bounded term list, or a documented deferral.	Rejecting it merely because no SPARQL query can yet answer it.
Validating	Test whether the ontology can answer an intended information need.	Query result, entailment, or constraint outcome over a declared fixture.	Calling a prose question executable without an expected result and reasoning configuration. Validating CQs should be formalizable and answerable (arxiv.org).
Foundational	Align a domain entity to a category in a foundational ontology.	A justified alignment decision, possibly with formal consistency or entailment checks.	Treating a philosophical modeling choice as a retrieval query. QuO connects this type to the alignment process (arxiv.org).
Relationship	Determine arity, participants, elementary status, or properties of a relation (arxiv.org).	Modeler decision plus axioms and targeted checks where formalization is meaningful.	Assuming every relation-design question has an informative query answer. Some require modeler judgment.
Metaproperty	Classify an entity through properties such as rigidity or identity criteria.	Expert analysis recorded with the model commitment and any formal consequences.	Conflating classification evidence with an instance-data lookup. QuO describes it as classifying a foundational or domain entity (arxiv.org).

The taxonomy is functional, not merely grammatical. Two questions beginning with “Which” can belong to different types because their intended use differs. Classification should therefore precede formalization. It determines whether the next artifact should be an executable test, a structured review, or a documented scope decision.

From a business need to a precise competency question

The first drafting move is not “write a query.” It is “name the decision.” A stakeholder question belongs in ontology scope when its answer changes an action, judgment, or design commitment for an identified actor, and when representing the required knowledge is plausibly the ontology's responsibility.

Start with the decision record

Capture these fields before polishing the question:

- **Decision:** What choice will the answer support?
- **Actor:** Who makes or approves that choice?
- **Consequence:** What changes if the answer differs?
- **Frequency:** Is the need one-time, periodic, event-triggered, or continuous?
- **Ontology role:** Must the ontology represent terms, infer consequences, constrain data, or only define scope?
- **Evidence boundary:** Which sources and observations are permitted?

- **Escalation:** Who resolves ambiguity or changes the requirement?
- **Exit condition:** What observable answer or review outcome is sufficient?

Goal-modeling research supports linking CQs to organizational goals rather than collecting them as isolated prompts (periodicos.ufmg.br). NeOn's ontology requirements specification similarly treats CQ responses as a form of requirement specification and records answers and priorities (research-archive.stem.open.ac.uk).

For an executable candidate, the Industrial Ontologies Foundry process asks for sample answers and an account of how answers are expected to be obtained, and it requires associated validation data (oagi.atlassian.net). Without answer evidence, the item may still be a legitimate scoping question, but it is not yet an executable acceptance test.

Rewrite ambiguity before formalization

“Which suppliers are strategic?” is not yet testable. It leaves strategic status, supplier identity, effective date, organizational viewpoint, and evidence source undefined. Rewrite each unstable dimension:

- **Quantifier:** Replace “all,” “any,” “often,” or “significant” with an explicit threshold or a named policy term.
- **Identity unit:** State whether names, records, legal entities, facilities, products, or roles are compared.
- **Time:** State observation date, effective interval, event time, and whether historical answers are required.
- **Jurisdiction or policy scope:** Name the governing geography, organizational unit, policy version, or taxonomy.
- **Source scope:** Identify permitted datasets, documents, graphs, and import closure.
- **Unknown handling:** Distinguish not found, blocked, not applicable, contradicted, and false.
- **Answer form:** Specify a Boolean, binding set, count, classification, explanation, or review decision.
- **Ordering and tolerance:** Define ranking, tie behavior, numeric tolerance, and duplicate treatment.

This is requirements work, not syntactic cleanup. The dataset analysis found that missing vocabulary can prevent construction of the expected query, and it also identified vague wording among the causes of nontranslation (arxiv.org). A failed formalization may therefore expose a deficient question, an ontology-scope gap, or both.

Apply a quality rubric

A reviewable CQ should pass six tests:

- **Bounded terms:** Every decision-relevant noun and predicate has an agreed meaning.
- **Explicit scope:** Identity, time, policy, source, and graph boundaries are declared.
- **Typed answer:** The expected form is known before implementation.
- **Falsifiability:** At least one possible result would show the requirement is not met.
- **Counterexample:** A plausible near miss tests the intended boundary.
- **Owner sign-off:** One accountable role can accept, reject, or supersede the contract.

The rubric does not require premature formal precision. A scoping CQ may pass through an expert-review protocol. What it cannot do is float without a decision, answer criterion, or owner.

The traceability contract

A natural-language question is compact but lossy. The traceability contract expands it into a maintained requirements object. It should receive a stable identifier even if its wording, query, or fixture later changes. That mirrors general requirements practice in which each requirement has an identifier and definitive source (www.nasa.gov). ISO/IEC/IEEE 29148 defines requirements management as maintaining and tracing requirements through the life cycle, and defines a traceability matrix as linking them upward to needs and downward to implementation (www.iso.org).

Table 2 gives the minimum contract. Teams may add risk, priority, cost, or dependency fields, but should not remove the links needed to reproduce a decision.

Contract field	Required content	Acceptance question
Identifier and status	Stable ID; draft, approved, executable, deferred, rejected, superseded, or retired.	Can an old report still identify the requirement it used?
Decision and owner	Decision supported, accountable role, consequence, approval date.	Is there a named action and a person or role authorized to change it?
Question and type	Versioned natural-language text and one of the five QuO types, with rationale.	Does the classification match the question's purpose?
Scope	Terms, identity unit, time, policy or jurisdiction, source and graph boundaries.	Would two implementers select the same population and interval?
Permitted evidence	Source classes, observation dates, provenance requirements, and disallowed evidence.	Can every expected value be defended from allowed evidence?
Formalization	SPARQL, entailment assertion, SHACL shape, expert protocol, or deferred-research plan.	Does the verification form test the claim actually made?
Reasoning regime	Explicit triples, RDF Schema, OWL profile, imports closure, rules, reasoner and materialization assumptions.	Could a different engine or closure change the answer?
Fixture and version	Positive, negative, boundary, and unanswerable cases with provenance.	Can the exact test state be reconstructed?
Expected result	Boolean, bindings, graph, count, constraint report, or signed review outcome, including comparison rules.	Is pass or fail decidable without inspecting implementation details?
Counterexample	Smallest plausible case that should not satisfy the requirement.	Does the contract reject a near miss?
Result and cause	Actual output, added or removed answers, runtime configuration, and diagnosed cause.	Can a reviewer separate ontology, fixture, reasoner, and requirement changes?
Review trigger	Scheduled date or event, source change, policy change, ontology release, or owner decision.	When must the contract be revisited or retired?

The contract should preserve semantics without freezing accidental implementation details. An ontology-focused lifecycle specification makes a complementary distinction: acceptance does not imply review approval, and approval does not prove an expected answer true (ontosome.com). It also treats a fixture as input evidence, not a hand-authored declaration that a query passed (ontosome.com).

Expected bindings should be compared as RDF terms or result sets, not as row order unless ordering is part of the decision. OWLunit compares supplied-query output with the expected result using graph isomorphism, and accepts expected results encoded in JSON or the W3C result-set vocabulary (github.com). That is stronger than comparing screenshots, but the contract must still say whether blank-node renaming, ordering, datatype normalization, or numeric tolerance matters.

Verification forms, fixtures, and expected answers

Verification should be selected from the required answer, not from tool familiarity. SPARQL supplies several result forms, but they have different semantics. **SELECT** returns variables and bindings, while **ASK** reports only whether at least one solution exists (www.w3.org). A contract that expects a complete population should not be reduced to ASK merely because one positive instance exists.

Choose among five verification forms:

- **SPARQL result set:** Use when the acceptance evidence is a defined set of resources, literals, aggregates, or ordered bindings.
- **ASK or entailment:** Use when acceptance is the existence of a solution or a specific logical consequence under a named regime.
- **SHACL validation:** Use when the requirement is a constraint on a data graph and the desired evidence is a validation report.
- **Expert review:** Use for scope, foundational alignment, relationship design, or metaproperty judgments that are not reducible to retrieval.
- **Deferred research:** Use when needed evidence, vocabulary, ownership, or formalization is unavailable. Record why and what would unblock it.

The reasoning regime is part of the expected answer. SPARQL itself does not mandate inference (jena.apache.org). The W3C defines distinct RDF, RDF Schema, datatype, OWL, and Rule Interchange Format entailment regimes, and OWL Direct Semantics requires the active graph to correspond to an OWL 2 DL ontology (www.w3.org). OWL API reasoners operate over an ontology's imports closure, so that closure belongs in the reproducible configuration (owlcs.github.io).

SHACL answers a different question. Validation takes a data graph and shapes graph and returns a report; the Recommendation makes no assumption that all entailed triples are already materialized (www.w3.org). A SHACL failure may therefore depend on whether inference was run before validation. The contract must say so.

The selected form should also state how errors, timeouts, empty results, and unsupported inference are recorded instead of being misread as negative answers.

Fixtures and worked examples

- **Positive fixture:** The smallest case that should return the expected answer.
- **Negative fixture:** A superficially similar case that should not return it.
- **Boundary fixture:** A case exactly at a time, cardinality, policy, or identity boundary.

- **Unanswerable fixture:** A case with absent, blocked, contradictory, or out-of-scope evidence.

Under OWL's open-world assumption, a missing assertion may be true but absent, rather than false (www.w3.org). Consequently, the unanswerable fixture is not an optional edge case. It verifies that “unknown” is not silently converted into “no.”

Table 3 follows three related questions through the method. The example is hypothetical, as required, but its verification choices follow the cited standards.

Worked question (Hypothetical Example)	Classification and status	Fixture and verification	Expected evidence
CQ-A: Which organizations controlled Facility F on observation date D, according to permitted corporate records?	Validating, **executable**. Identity unit, control relation, observation date, and source scope are explicit.	Positive and boundary fixtures; SPARQL SELECT under the declared dataset and entailment regime. A SPARQL dataset has a default graph and may have named graphs (www.w3.org).	An unordered set of canonical organization identifiers, each linked to permitted evidence, plus explicit unknown status if access was blocked.
CQ-B: Must the `controls` relation be modeled directly or derived from more elementary relations?	Relationship, **expert-reviewed** and partly testable.	Two competing axiom designs, consistency and entailment checks, and a signed modeler review. QuO says relationship CQs determine key relation characteristics (arxiv.org).	A documented modeling decision, rationale, affected axioms, and regression tests for observable consequences. No invented SELECT result.
CQ-C: Which source is most authoritative for every possible ownership dispute?	Scoping, **rejected as written**. It contains an unbounded universal and undefined authority criterion.	No executable fixture. Rewrite into decision-specific source rules or defer to policy research.	Rejection reason, owner, and a review trigger. The natural-language question is retained so the discarded assumption remains visible.

The table demonstrates why “not executable” is a legitimate status rather than a failure. What matters is that the question's role and disposition remain visible. An executable query without CQ-B's design context could pass after an unintended modeling change. A natural-language CQ-A without its fixture and expected binding rules could be interpreted several incompatible ways.

Regression, change, and retirement

A contract becomes most valuable after the first successful run. Every ontology change should re-execute affected contracts and produce a diff at the semantic level. For SELECT, compare added and removed bindings. For ASK or entailment, compare truth status. For SHACL, compare focus nodes, paths, severities, and result messages. For expert review, record the new decision and the evidence considered.

Diagnose a changed result against four causes:

- **Ontology change:** Vocabulary, axioms, imports, annotations, or mappings changed.
- **Fixture change:** Test data, source snapshot, graph selection, or identity assumptions changed.
- **Reasoner change:** Engine, profile, rules, materialization, or imports closure changed.

- **Requirement change:** The owner changed the decision, scope, answer criterion, or acceptable evidence.

OWLunit usefully separates CQ verification from inference verification. Its inference tests can compare expected and actual inferences, while CQ verification can compare query results to an expected result (github.com). RDF4J also supports declaring shapes once and validating multiple datasets, which is useful for controlled fixture families (rdf4j.org). These capabilities do not diagnose causality automatically. The contract's version links do that work.

A regression diff should report:

- **Baseline:** Contract, ontology, fixture, query or shape, reasoner, and source versions.
- **Expected:** Prior accepted bindings, entailment state, or validation outcome.
- **Actual:** Current output in the same comparison form.
- **Delta:** Added answers, removed answers, status changes, and comparison tolerances.
- **Cause:** Ontology, fixture, reasoner, requirement, or unresolved.
- **Disposition:** Fix implementation, update expectation, supersede contract, or accept documented change.
- **Approver:** Owner and review date.

SEBoK identifies change-impact assessment as a principal use of traceability and recommends defining verification success criteria, strategy, and method when requirements are generated (sebokwiki.org). These practices support diagnosing a diff before updating its baseline.

Retirement should never mean deletion. W3C PROV defines revision as creating an entity that is a revised version of an original, and its invalidation concept marks an entity as no longer available for use (www.w3.org). Mark a CQ **superseded** when another contract replaces its decision meaning, **retired** when the decision no longer exists, **rejected** when it never became an accepted requirement, and **deferred** when evidence or formalization remains unavailable. Bidirectional traceability exists partly to identify which tests and other artifacts a requirement change affects (swehb.nasa.gov). Preserving the earlier state makes later acceptance evidence intelligible.

Data Analysis and Evidence

The available datasets quantify translation difficulty, but they should not be read as universal success rates. The **234-question** corpus came from several ontologies, domains, author groups, and development settings, and produced **131 formalized SPARQL-OWL queries** (pubs.cs.uct.ac.za). The descriptive ratio is **131 / 234 = 56.0%**, rounded to one decimal. It is a property of that dataset and method, not a claim that 44.0% of all ontology CQs are inherently impossible.

The same study found **106 distinct linguistic CQ patterns** and **46 query signatures** (arxiv.org). The nine most common query patterns accounted for **63.1%** of its query set, yet natural-language and formal patterns had a many-to-many relationship (arxiv.org). This is evidence for reusable templates, but evidence against selecting a query shape solely from surface wording.

BigCQ demonstrates scale and the same caution. Its official repository reports **77,575 unique synthetic CQ templates** and **549 unique SPARQL-OWL query templates** (github.com). The associated paper reports that one uncovered query-template form represented **19.38%** of the earlier corpus's queries (arxiv.org). A larger synthetic count therefore does not prove empirical coverage.

The QuO evidence measures variety rather than translation. The original ROCQS release contained **438 CQs** across five types (arxiv.org). The precursor conference chapter was published online on **20 November 2024**, and the later workshop volume was published on **2 March 2026** (link.springer.com) (ceur-ws.org). As of September 2026, the evidence supports a five-type working taxonomy, not a universal benchmark for how often each type occurs in every organization.

Three calculations follow:

- **Observed formalization share:** 131 divided by 234 equals **56.0%** for the published corpus (pubs.cs.uct.ac.za).
- **Concentration among formal patterns:** the top nine of 46 query patterns, **19.6%** of pattern types, covered **63.1%** of queries (arxiv.org).
- **Synthetic expansion:** 77,575 CQ templates divided by 549 query templates is about **141.3 CQ templates per query template**, a scale descriptor only, not a translation probability (github.com).

These numbers justify investing in pattern libraries while retaining question-specific review. They also justify an explicit translation status field. A program should publish raw counts of **executable**, **partly executable**, **expert-reviewed**, **deferred**, and **rejected** contracts, with reasons, rather than a single pass percentage that mixes unlike question types.

Implications and Future Directions

The traceability-contract approach changes ontology governance in four ways. First, it makes the sponsor's decision part of the technical artifact. Second, it prevents scoping and foundational questions from being mislabeled as failed queries. Third, it makes test outcomes reproducible by fixing the ontology, fixture, evidence, graph, and reasoning context. Fourth, it provides an explicit retirement history.

Reliable Context is relevant where CQ evidence must remain defensible beyond a test run. Its first-party guarantee is one canonical record per real thing, with a cited, verbatim, dated source behind every value and declared unknowns rather than blanks. The same page states that past states are reproducible and that exports carry source, excerpt, locator, and dates. Those observable properties can support a contract's evidence and replay requirements. They do not replace an ontology editor, reasoner, SPARQL engine, SHACL processor, or test runner.

Several implementation priorities follow:

- **Manage contracts as versioned records:** Keep the question, owner, decision, evidence boundary, formalization, fixture, and result linked across releases.
- **Separate evidence layers:** Preserve source observations separately from inferred consequences and validation outcomes.

- **Make unknown a result:** Report not found, blocked, not applicable, contradiction, and out of scope distinctly.
- **Test reasoning explicitly:** Run direct-triple and entailed-answer variants when both matter, and name the imports closure.
- **Audit translation status:** Count what is executable and what requires human judgment without treating the latter as defects.
- **Review drift:** Trigger review when sources, policies, ontology terms, reasoners, or stakeholder decisions change.

Reliable Context's public guarantees illustrate the evidence layer needed when values leave the ontology team. It says every value has a verbatim cited source and observation date, unknowns are declared with reasons, and past states can be reproduced. For organizations that must defend fused private and public information, these properties complement ontology acceptance testing. A team that only needs graph storage, vector retrieval, master-data management, a data catalog, or conversational answers should use tools built for those jobs.

Research should next evaluate contracts end to end. Useful measures include inter-reviewer agreement on CQ type, change-impact precision, false pass and false fail rates, time to diagnose a regression, and the share of answers whose evidence remains reproducible after source change. Passing CQs can provide initial validation, but published requirements work cautions that planned functionality and stakeholder sufficiency still require further evaluation (link.springer.com).

Frequently Asked Questions (FAQs)

What are competency questions in ontology engineering?

They are natural-language questions that state or probe the knowledge an ontology is intended to cover. The original enterprise-modeling method introduced them as a basis for defining ontology scope (eil.mie.utoronto.ca). Depending on type, they may guide scope, validate answers, support foundational alignment, analyze relations, or test metaproperties.

How should ontology competency questions be written?

Begin with the decision, actor, consequence, and owner. Then bound quantifiers, identity, time, policy or jurisdiction, evidence sources, graph scope, unknown handling, and answer form. Add a counterexample and review trigger. Formalize only after classifying the CQ's purpose.

Are all competency questions SPARQL queries?

No. The five-type taxonomy includes purposes other than executable validation. The 234-question dataset produced 131 SPARQL-OWL formalizations, with multiple reasons for the gap (pubs.cs.uct.ac.za). An ontology-focused specification likewise states that not every foundational or stakeholder question can safely be reduced to a graph query (ontosome.com).

How does SPARQL validate an ontology competency question?

A declared query runs over a versioned fixture and named reasoning regime. Its ASK result, SELECT bindings, or graph output is compared with a predeclared expected result. OWLunit documents vocabulary checks and isomorphic expected-result comparison for CQ verification (github.com). This validates the declared test configuration, not universal completeness.

When should SHACL be used instead?

Use SHACL when the acceptance claim is a constraint over a data graph, such as required cardinality, datatype, or permitted value patterns. SHACL validation takes data and shapes graphs and produces a validation report (www.w3.org). It does not by itself answer every ontology-level entailment or scope question.

What is ontology requirements traceability?

It is the maintained ability to follow an ontology requirement backward to its stakeholder need and evidence, and forward to modeling decisions, tests, results, and later changes. NeOn explicitly linked competency questions to derived concepts to improve traceability of concept origins (research-archive.stem.open.ac.uk).

Does passing every fixture prove ontology completeness?

No. It shows that the declared ontology and data, under the declared query and reasoning configuration, met finite expectations. Open-world semantics means absence may still be unknown rather than false (www.w3.org). Real-world scope, source coverage, stakeholder sufficiency, and future cases require separate evidence.

Conclusion

Competency questions are most useful when they remain connected to both sides of ontology engineering: the human decision that creates the requirement and the technical evidence that later supports acceptance. A prose question alone cannot preserve that connection. A query alone cannot explain it.

The traceability contract supplies the missing durable unit. It names the owner and decision, classifies the question, bounds identity, time, source, and unknowns, selects a verification form, versions the fixture and reasoning regime, states the expected answer and counterexample, and records change and retirement. That structure allows early scoping questions to remain useful before an ontology exists and allows executable validating questions to become repeatable regression evidence later.

The empirical record supports disciplined pluralism. Different CQ types serve different purposes, and one published dataset achieved only partial formalization with a many-to-many mapping between linguistic and formal patterns. Large synthetic libraries expand reusable material without establishing universal coverage. The correct management response is not to force every question into SPARQL. It is to expose translation status and preserve the reason.

For sponsors, the acceptance rule is straightforward: no question enters scope without a decision and owner. For domain experts, ambiguous quantifiers, identity, time, evidence, and unknowns must be resolved or recorded. For knowledge engineers, every executable test needs a versioned fixture, explicit reasoning

assumptions, expected result, and counterexample. When those elements remain linked across change, a competency question becomes more than documentation. It becomes defensible evidence of what the ontology was required to do, what was tested, what happened, and why the result was accepted.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://eil.mie.utoronto.ca/theory/enterprise-modelling/entmethod
2. https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
3. https://arxiv.org/html/2412.13688
4. https://research-archive.stem.open.ac.uk/stemra-data/uploads/sites/136/2025/03/NeOn_2008_D5.4.1.pdf
5. https://pubs.cs.uct.ac.za/id/eprint/1394
6. https://arxiv.org/html/1811.09529
7. https://www.w3.org/TR/sparql11-query
8. https://www.w3.org/TR/owl2-primer
9. https://link.springer.com/article/10.1007/s00766-020-00335-z
10. https://lot.linkeddata.es
11. https://www.nasa.gov/reference/system-engineering-handbook-appendix
12. https://swehb.nasa.gov/spaces/7150/pages/16449675/SWE-059%2B-%2BBidirectional%2BTraceability%2BBetween%2BSoftware%2BRequirements%2Band%2BSoftware%2BDesign
13. https://periodicos.ufmg.br/index.php/jidm/article/view/124
14. https://oagi.atlassian.net/wiki/spaces/IOF/pages/4703289348/Proposed%2BOntology%2BDevelopment%2BProcess%2BV1.0
15. https://www.iso.org/obp/ui?_escaped_fragment_=iso:std:iso-iec-ieee:29148:ed-2:v1:en
16. https://ontosome.com/modules/competency-question/spec
17. https://github.com/luigi-asprino/owl-unit
18. https://jena.apache.org/tutorials/sparql.html
19. https://www.w3.org/TR/2013/REC-sparql11-entailment-20130321
20. https://owlcs.github.io/owlapi/apidocs_5/org/semanticweb/owlapi/reasoner/OWLReasoner.html
21. https://www.w3.org/TR/shacl
22. https://rdf4j.org/documentation/programming/shacl
23. https://sebokwiki.org/wiki/System_Requirements
24. https://www.w3.org/ns/prov
25. https://github.com/dwisniewski/BigCQ
26. https://arxiv.org/html/2105.09574
27. https://link.springer.com/chapter/10.1007/978-3-031-77792-9_8
28. https://ceur-ws.org/Vol-4176

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.