A Requirement Interpretation Agent for Formalizing Quality Requirements in Enterprise Knowledge Graphs*

Narciso Arruda[1,2,*,†], V. M. Pequeno[3,†], Flávio Lima[2,†], Frederico Ribeiro[2,†], V. M. P. Vidal[1,†] and M. A. Casanova[4,†]

[1]*Departamento de Computação, Federal University of Ceará, Fortaleza, Ceará, Brazil*
[2]*TSA Lynx Process – Belo Horizonte, MG – Brazil*
[3]*Escola Superior Náutica Infante D. Henrique – Portugal*
[4]*Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Brazil*

Abstract

Data quality is commonly understood as fitness for use, meaning that quality expectations depend on the intended application, user needs, and context of use. In Enterprise Knowledge Graphs (EKGs), users and domain experts often express such expectations as natural-language quality criteria, whereas computational Data Quality Assessment (DQA) approaches require formal artifacts, such as quality dimensions, metrics, scopes, thresholds, and executable assessment procedures. This creates a gap between user-defined quality needs and machine-actionable quality assessment. This paper proposes a metadata-driven Requirement Interpretation Agent for formalizing quality requirements over EKG semantic views. The agent retrieves contextual metadata from the EKG metadata graph and maps a natural-language criterion to a formal quality requirement specifying the target view, quality dimension, applicable quality metrics, semantic scope, evaluation level, comparison operator, and threshold. The resulting requirement is represented as RDF metadata by reusing existing quality vocabularies, such as DQV, and by introducing lightweight requirement-specific terms when needed. We evaluate the approach on 140 synthetic criteria across Exported, Linkset, Unification, and Fusion Views. The full configuration achieves 98.6% status accuracy, 90.6% core-requirement accuracy, and 100% expected-metric precision at 52.3% coverage. Removing retrieved quality metadata reduces core accuracy to 27.1% and grounding validity to 3.0%, whereas removing retrieved view metadata reduces them to 53.1% and 77.9%, respectively. These results indicate that the two metadata components play complementary roles in producing grounded, reusable quality requirements.

Keywords

Enterprise Knowledge Graphs, Data Quality Assessment, Quality Requirements, Semantic Views, Knowledge Graph Validation, Large Language Models

CEUR-WS.org/Vol-4322/paper_23.pdf

1. Introduction

Data quality is commonly understood as fitness for use, meaning that quality expectations depend on the intended application, the user's needs, and the context in which data will be consumed. In Enterprise Knowledge Graphs (EKGs), these expectations are often expressed by users and domain experts as natural-language quality criteria, whereas computational Data Quality Assessment (DQA) approaches require formal artifacts, such as metrics, constraints, thresholds, scopes, and executable assessment procedures. Existing DQA approaches for EKGs largely assume that quality requirements are already available in a formal, machine-actionable form [1, 2], leaving the process of eliciting and formalizing such requirements from user-defined criteria comparatively underexplored.

This paper addresses this gap in the context of EKG semantic views. Following Vidal et al. [3], we use the term *semantic view* to refer to an ontology-mediated, integrated representation of an organization's heterogeneous data sources within an EKG. In the DDP-SV framework [4], a semantic

DMKG'26: 2nd International Workshop on Data Management for Knowledge Graphs, October 2026, Bari, Italy
*Corresponding author.
†These authors contributed equally.
narcisoarruda@gmail.com (N. Arruda); valeriapequeno@enautica.pt (V. M. Pequeno); flavio.lima@lynxprocess.com (F. Lima); frederico.ribeiro@lynxprocess.com (F. Ribeiro); vvidal@lia.ufc.br (V. M. P. Vidal); casanova@inf.puc-rio.br (M. A. Casanova)
0000-0003-3873-8468 (N. Arruda); 0000-0002-6424-0252 (V. M. Pequeno); 0000-0003-0765-9636 (M. A. Casanova)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

view is constructed through different types of views, including exported, linkset, unification, and fusion views (detailed in Section 2). Assessing the quality of these views requires considering the target view, view type, ontology terms, evaluation scope, available evidence, and quality metadata. Since such metadata is represented using RDF vocabularies and ontologies, it can provide semantic context to guide the interpretation of user-defined quality criteria.

We propose a *Requirement Interpretation Agent* that transforms natural language quality criteria into formal quality requirements grounded in the metadata graph of an EKG. The agent uses an LLM to interpret the user criterion in the context of retrieved view specifications and quality metadata, identifies the target view, quality dimension, applicable quality metrics, semantic scope, evaluation level, comparison operator, and threshold. The resulting requirement is represented as RDF metadata by reusing existing quality vocabularies, such as DQV, and by introducing lightweight requirement-specific terms only when necessary.

From a knowledge graph data-management perspective, this representation introduces an explicit requirement layer for quality-aware validation of EKG semantic views. Requirements can be persisted, queried, linked to provenance information, and consumed by downstream mechanisms such as SPARQL template instantiation, DQV-based quality measurement, and SHACL-like validation procedures. The language model, therefore, supports natural-language understanding and semantic alignment, while the concepts selected during interpretation are validated against the view specifications and quality metadata available in the EKG metadata graph. This separation allows the formalized requirement to be reused without requiring the original natural-language criterion to be reinterpreted for each downstream mechanism.

The main contribution of this paper is a metadata-driven interpretation layer that bridges user-defined quality criteria and machine-actionable DQA artifacts for EKG semantic views. Rather than treating the problem as direct natural-language-to-RDF generation, the proposed approach combines LLM-based interpretation with explicit grounding in view and quality metadata and validation of the resulting formal requirement. More specifically, it combines: (i) a formal model for representing the target view, quality dimension, selected metrics, semantic scope, evaluation level, operator, and threshold; (ii) metadata-grounded interpretation and validation across four semantic-view types; and (iii) an RDF representation that supports traceability and reuse by downstream quality-assessment mechanisms. The scope of this paper is the interpretation and formalization of quality requirements. The execution of quality measurements, assessment of requirement satisfaction, diagnosis, and recommendation are downstream activities and are not experimentally evaluated here. A running example involving the coverage of Wikipedia's primary-topic pages for MusicBrainz musical artists illustrates how an informal criterion is transformed into a structured and reusable quality requirement.

The remainder of this paper is organized as follows. Section 2 introduces the background concepts underlying our proposal, including semantic views, the DDP-SV framework, and EKG metadata graphs. Section 3 presents the formal quality requirement model. Section 4 describes the proposed Requirement Interpretation Agent and its metadata-grounded interpretation process. Section 5 reports the experimental evaluation. Section 6 reviews related work, positioning our approach with respect to existing DQA and quality-requirement formalization strategies. Finally, Section 7 concludes the paper and outlines directions for future work.

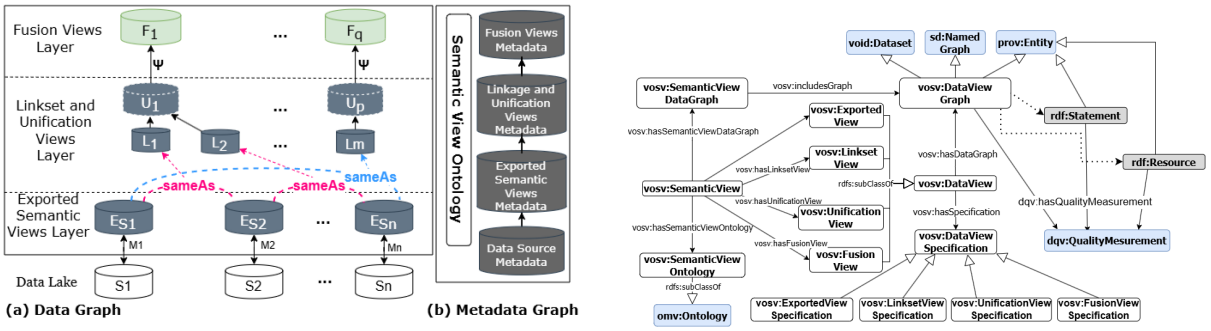
2. Semantic Metadata Context for EKG Data Quality Requirements

This section introduces the DDP-SV framework and VoSV metadata used to ground quality requirements.

2.1. Four-Layer Architecture for Constructing EKG Semantic Views

The DDP-SV framework provides a structured approach for organising both the data graph and the metadata graph of an EKG semantic view. As illustrated in Figure 1a, a semantic view comprises two interconnected knowledge graphs: a *data graph*, which represents the integrated content; and a *metadata graph*, which captures structural, provenance, and quality information. The data graph,

shown in Figure 1a(a), is organised into a four-layer hierarchical architecture, in which each layer corresponds to a distinct stage of semantic integration:



(a) Data Design Pattern for EKG's Semantic Views (b) Vocabulary of Semantic Views (VoSV)

Figure 1: Data design pattern and vocabulary for EKG semantic views.

- **Exported Views Layer**: Contains RDF views generated by mapping raw data sources to a shared vocabulary defined in the Semantic View Ontology (SVO), ensuring semantic consistency.
- **Linkset Views Layer**: Establishes identity links (e.g., `owl:sameAs`) between equivalent entities across exported views, enabling semantic alignment and cross-referencing.
- **Unification Views Layer**: Integrates semantically equivalent entities into canonical representations, consolidating references to the same real-world objects.
- **Fusion Views Layer**: Resolves conflicts among unified entities by applying resolution strategies to produce a consistent and enriched view.

The semantic view is constructed progressively from ontology modeling and exported views to linkset, unification, and fusion views. The associated metadata graph records the view specifications, their relationships, and the corresponding structural, provenance, and quality metadata. This metadata layer provides the semantic context required by the requirement-grounding process proposed in this paper.

2.2. Modeling Semantic View Metadata with VoSV

The DDP-SV framework adopts the Vocabulary of Semantic Views (VoSV) to represent semantic-view metadata in a structured and interoperable form. As illustrated in Figure 1b, VoSV connects the semantic view to its data sources, view specifications, data graphs, and quality measurements. By making these elements explicit and machine-actionable, the metadata graph provides the contextual information that grounds quality requirements in our approach.

3. Formal Quality Requirement Model

This section introduces the model used to transform user-defined quality criteria into formal and machine-actionable quality requirements. The model specifies the quality condition to be verified, its semantic scope, and the quality metrics used to operationalize it. The resulting requirement is represented as RDF metadata to support traceability, validation, and reuse by different Data Quality Assessment (DQA) mechanisms.

Consider the running criterion:

> "At least 90% of MusicBrainz musical artists must have a Wikipedia primary-topic page."

Although understandable to a user, this criterion is not directly executable. Its formalization requires grounding *MusicBrainz musical artists* and *Wikipedia primary-topic page* in the class and property

available in the target view, identifying the applicable quality dimension and metrics, and normalizing *at least 90%* into the comparison operator $\geq$ and the threshold 0.9. These elements are determined using the target-view specification and the quality metadata available in the EKG metadata graph.

3.1. Formal Definition

A *formal quality requirement* is a machine-actionable condition specifying the required quality level of a target view for a defined evaluation scope. It is represented as:

$$RQ = \langle v, d, \mathscr{M}, C, \mathscr{P}, l, op, t \rangle, \tag{1}$$

where v is the target view, d is the quality dimension, $\mathscr{M}$ is a non-empty set of metric selections, C is an optional target class, $\mathscr{P}$ is an optional set of target properties, l is the evaluation level, op is a comparison operator, and t is the required threshold.

Each metric selection associates a quality metric with the operational role that it assumes in the requirement:

$$\mathscr{M} = \{\langle m_i, r_i \rangle \mid r_i \in \{\text{observed}, \text{expected}\}\}, \qquad 1 \leq |\mathscr{M}| \leq 2. \tag{2}$$

An observed metric is computed over the materialized target-view data, whereas an expected metric is computed over the view specification or its metadata. At least one metric must be selected for a valid requirement. The current model supports at most one metric for each operational role. When observed and expected metrics are selected together, their joint use must be supported by the retrieved quality metadata:

$$\begin{aligned} &\langle m_o, \text{observed} \rangle \in \mathscr{M} \wedge \langle m_e, \text{expected} \rangle \in \mathscr{M} \\ &\qquad \Rightarrow \text{Compatible}(m_o, m_e, v, C, \mathscr{P}, l, op, t). \end{aligned} \tag{3}$$

Compatibility requires the metrics to address the same quality aspect and to be compatible with respect to the target view, semantic scope, evaluation level, measurement scale, preference direction, reference population, and mathematical interpretation.

The elements C and $\mathscr{P}$ define the semantic scope of the requirement. Their presence depends on the evaluation level and the target-view type. For example, an attribute-level requirement may specify a target class and property, whereas a view-level requirement may require only the target view.

For this requirement, the observed condition is evaluated as:

$$\text{Satisfied}(RQ_1, m_o) \iff m_o(\texttt{svm:EV_MusicBrainz_Artist}, \texttt{mo:MusicArtist}, \{\texttt{foaf:isPrimaryTopicOf}\}) \geq 0.9. \tag{4}$$

The specification-level condition is evaluated as:

$$\text{Attainable}(RQ_1, m_e) \iff m_e(\texttt{svm:EV_MusicBrainz_Artist}, \texttt{mo:MusicArtist}, \{\texttt{foaf:isPrimaryTopicOf}\}) \geq 0.9. \tag{5}$$

The observed metric determines whether the materialized exported view satisfies the required Wikipedia primary-topic coverage. The expected metric provides complementary evidence about whether the current view specification contains sufficient mapping support for the same class and property.

For the running example, the requirement is instantiated as:

$$\begin{aligned} RQ_1 = \langle &\texttt{svm:EV_MusicBrainz_Artist}, \texttt{qv:Completeness}, \\ &\{\langle \texttt{qv:AttributeCoverage}, \text{observed} \rangle, \langle \texttt{qv:PropertyMappingCompleteness}, \text{expected} \rangle\}, \\ &\texttt{mo:MusicArtist}, \{\texttt{foaf:isPrimaryTopicOf}\}, \\ &\texttt{qv:AttributeLevel}, \geq, 0.9 \rangle. \end{aligned} \tag{6}$$

Here, `qv:AttributeCoverage` is selected as the observed metric. It measures the proportion of `mo:MusicArtist` instances in the materialized target view that contain a value for `foaf:isPrimaryTopicOf`. This property links a musical artist to a Wikipedia resource for which the artist is the primary topic.

The metric `qv:PropertyMappingCompleteness` is selected as the expected metric. It evaluates whether the exported-view specification contains the mapping components required to generate `foaf:isPrimaryTopicOf` values for `mo:MusicArtist` resources.

Both metrics operationalize the same user-defined quality condition over different evidence sources. They are applicable to the same target view, quality dimension, evaluation level, class, and property, and their results use the same ratio scale, preference direction, comparison operator, and threshold. The observed metric evaluates the materialized view, whereas the expected metric evaluates the corresponding capability represented in the view specification.

3.2. RDF Representation and Reuse

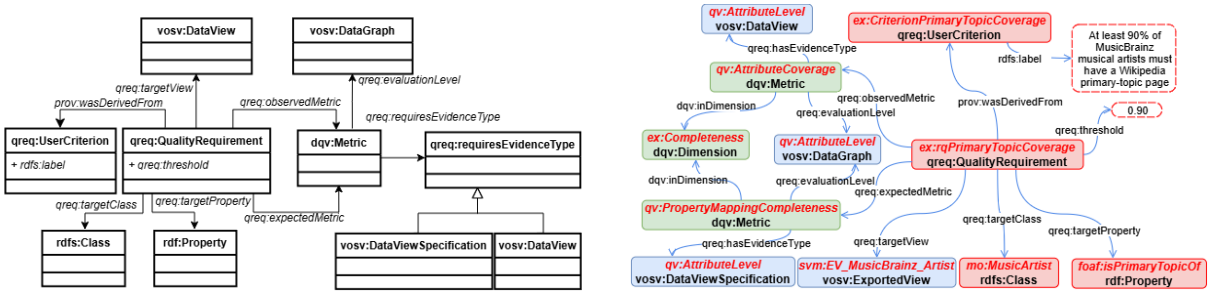
Formal quality requirements are represented as RDF metadata and stored in the EKG metadata graph. The representation follows a reuse-first strategy: existing vocabularies are adopted when their semantics are suitable, while a lightweight requirement vocabulary is introduced only when necessary.

Representing the requirement as RDF separates the interpretation of the requirement from its subsequent execution. Rather than translating a user criterion directly into an execution-specific artifact, such as a SPARQL query or a SHACL shape, the interpretation process produces a persistent, machine-actionable description of the user's quality intent. This description preserves the target view, quality dimension, selected metrics, semantic scope, evaluation level, operator, and threshold independently of the downstream mechanism that will consume it.

Four complementary vocabularies are used. The Vocabulary of Semantic Views, denoted by `vosv:`, represents the target view, its specification, and its materialized data graph. DQV provides the generic classes and properties for representing quality dimensions, metrics, and measurements. The quality vocabulary, denoted by `qv:`, defines the concrete quality dimensions and metrics applicable to semantic views. Finally, the requirement vocabulary, denoted by `qreq:`, represents the original user criterion, the formal requirement, its semantic scope, and the metrics selected to operationalize it.

The classes `qreq:QualityRequirement` and `qreq:UserCriterion` represent the formal requirement and its original natural-language criterion, respectively. The properties `qreq:targetView`, `qreq:targetClass`, and `qreq:targetProperty` define the evaluation scope. Selected metrics are linked to the requirement through `qreq:observedMetric` and `qreq:expectedMetric`, which encode the operational role assigned to each metric in the current requirement.

Figure 2 presents both the vocabulary fragment and its application to the running example.



(a) Fragment of the vocabulary for formal quality requirements.

(b) RDF representation of the Wikipedia primary-topic coverage requirement.

Figure 2: Vocabulary and RDF representation of the formal quality requirement.

In Figure 2b, the requirement targets `svm:EV_MusicBrainz_Artist`, `mo:MusicArtist`, and `foaf:isPrimaryTopicOf`. It is associated with the completeness dimension, the attribute-level evalu-

ation level, qv:AttributeCoverage as the observed metric, and qv:PropertyMappingCompleteness as the expected metric.

The RDF representation allows the requirement to be retrieved through SPARQL and used as input by different DQA mechanisms. Computed results can be linked to the requirement through dqv:QualityMeasurement, while provenance metadata can associate the formal requirement with the original criterion and the interpretation activity. The formal requirement, therefore, serves as a persistent, reusable contract between the user's quality intent and downstream DQA mechanisms. Query-template instantiation, validation-constraint generation, quality measurement, and subsequent agent-based assessment tasks can consume the same formalized requirement without requiring the original natural-language criterion to be interpreted again.

3.3. From Formal Requirements to Executable Artifacts

Once formalized as RDF, the requirement can be consumed by different execution mechanisms without repeating the natural-language interpretation step. The requirement specifies the parameters for predefined templates, while each downstream mechanism determines how those parameters are operationalized. For RQ_1, the target view, class, and property are used to instantiate a SPARQL query that computes qv:AttributeCoverage:

```
SELECT (xsd:decimal(?withProperty) /
        xsd:decimal(?total) AS ?coverage)
WHERE {
  { SELECT (COUNT(DISTINCT ?x) AS ?total)
    WHERE { GRAPH <VIEW> { ?x a <CLASS> . } } }
  { SELECT (COUNT(DISTINCT ?x) AS ?withProperty)
    WHERE { GRAPH <VIEW> {
      ?x a <CLASS> ; <PROPERTY> ?value .
    } } }
}
```

For RQ_1, <VIEW>, <CLASS>, and <PROPERTY> are instantiated with svm:EV_MusicBrainz_Artist, mo:MusicArtist, and foaf:isPrimaryTopicOf, respectively. The resulting coverage is compared with the operator and threshold stored in the requirement according to Equation 4.

The same semantic scope can also instantiate an instance-level SHACL shape that identifies resources without the required property. Such a shape provides diagnostic evidence but does not by itself enforce the population-level threshold. The aggregate threshold can be evaluated from the SPARQL result or through a SHACL-SPARQL constraint. Complete executable artifacts are available with the experimental material.

Thus, SPARQL and SHACL represent different operationalizations of the same formal requirement rather than independent interpretations of the user criterion. The intermediate RDF representation decouples the semantic interpretation of the requirement from the technology used to execute or validate it.

4. Requirement Interpretation Agent

The Requirement Interpretation Agent operationalizes the formal model introduced in Section 3. It transforms a natural-language quality criterion into a formal quality requirement grounded in the target view specification and in the quality metadata available in the EKG.

The agent follows a retrieval-augmented process. Authoritative RDF metadata is retrieved from the metadata graph, and semantic alignment and LLM-based reasoning are applied only to the retrieved candidates. Consequently, the agent can select only concepts and resources represented in the available EKG context.

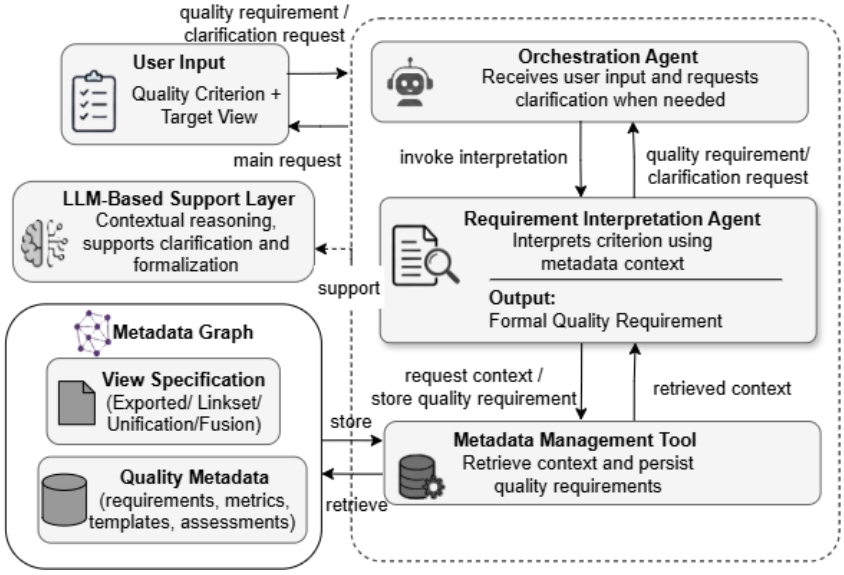


Figure 3: Metadata-driven architecture of the Requirement Interpretation Agent.

Table 1
Phases of the requirement formalization process.

Phase	Main operation	Output
Criterion analysis	Extracts entity and property mentions, quality intention, operator, and threshold.	Parsed criterion
Metadata grounding	Aligns user expressions with ontology terms and retrieves applicable quality metrics.	Candidate scope and metrics
Requirement construction	Combines the grounded concepts with the formal model defined in Section 3.	Candidate requirement
Validation	Checks the candidate against the view specification and quality metadata.	Validated requirement or non-valid status

The process produces either a validated formal requirement or a non-valid status indicating that clarification is required or that the criterion cannot be formalized using the available metadata.

4.1. Architecture and Interaction Flow

Figure 3 presents the metadata-driven architecture and interaction flow of the proposed agent.

The process starts when the user provides a natural-language quality criterion and identifies the target semantic view. The Orchestration Agent verifies the input and requests clarification when the criterion or target view is missing or ambiguous. Otherwise, it invokes the Requirement Interpretation Agent.

The agent obtains contextual information through the Metadata Management Tool, which retrieves two complementary types of metadata:

- *view specification metadata*, including the target view, view type, ontology terms, classes, properties, and other terms of the specification view; and
- *quality metadata*, including dimensions, metrics, evaluation levels, accepted operators and scales, and previously validated requirements.

Using this context, the agent performs the four phases summarized in Table 1.

For the running criterion, the expressions *MusicBrainz musical artists* and *Wikipedia primary-topic page* are aligned with `mo:MusicArtist` and `foaf:isPrimaryTopicOf`, respectively. The quantitative expression "at least 90%" is normalized into the operator $\geq$ and the threshold 0.9.

The retrieved quality metadata constrains metric selection based on the target view type, quality dimension, evaluation level, semantic scope, measurement scale, preference direction, and required evidence. In the running example, the agent selects `qv:AttributeCoverage` as the observed metric and `qv:PropertyMappingCompleteness` as the expected metric.

The selected elements are then combined to instantiate the formal requirement model. The resulting candidate corresponds to the requirement presented in Section 3 and must be validated before it is persisted or consumed by downstream DQA mechanisms.

The LLM-Based Support Layer assists with natural-language understanding, semantic alignment, candidate ranking, clarification generation, and explanation. However, authoritative concept retrieval, structural validation, and RDF persistence remain grounded in the metadata graph and deterministic rules.

4.2. Validation and Implementation

Validation checks whether the generated requirement is grounded in the retrieved metadata and compatible with the formal model. More specifically, it verifies:

- the validity of the target view and semantic scope;
- the applicability of the selected metrics to the quality dimension and evaluation level;
- the compatibility of the operator, threshold, scale, and preference direction; and
- the availability of the evidence required by each metric.

When observed and expected metrics are selected together, their correspondence is additionally checked with respect to the evaluated quality aspect, scope, measurement scale, preference direction, reference population, and mathematical interpretation. User confirmation may be requested when multiple plausible candidates remain or when this correspondence requires domain knowledge.

If validation succeeds, the requirement is serialized according to the RDF representation defined in Section 3.2 and persisted in the quality metadata graph. Otherwise, the agent returns a status such as `clarification_required`, `insufficient_context`, `unsupported_dimension`, or `validation_error`, without producing unsupported requirement metadata. The validation performed at this stage concerns the structural and semantic validity of the formal requirement itself; it does not evaluate whether the underlying data satisfies the requirement.

The agent is implemented as a hybrid component combining SPARQL-based metadata retrieval, semantic candidate ranking, LLM-supported requirement generation, deterministic validation, and RDF persistence. The generated structured output includes the target view, quality dimension, selected metrics, semantic scope, evaluation level, operator, threshold, and interpretation status. After validation and, when necessary, user confirmation, the Metadata Management Tool persists the requirement, its relation to the original criterion, the selected metrics, provenance information, and validation status.

4.3. Interactive Prototype

To support human inspection and manual validation, we implemented the *Requirement Agent Live Console*[1]. The interface allows users to select one of the four target semantic views, submit a natural-language quality criterion, and inspect the resulting formal requirement as a human-readable report. The prototype follows the architecture described in Section 4 and was used to qualitatively inspect a subset of the outputs produced during development.

5. Experimental Evaluation

This section evaluates the agent with respect to the criterion classification, core-requirement formalization, expected-metric selection, grounding validity, and operational cost.

[1]Available at https://tiny.cc/rq-agent-app.

5.1. Experimental Setup

The evaluation uses the `svm:DBB_Music_EKG` and four target views: `svm:EV_MusicBrainz_Artist`, `svm:LV_MusicArtist`, `svm:UV_MusicArtist`, and `svm:FV_MusicArtist`. Each case contains a natural-language criterion, a target-view descriptor, and metadata retrieved from the EKG metadata graph.

The test set contains 140 synthetic criteria, with 35 cases per view: 24 valid, eight ambiguous, and three rejected. Canonical requirements were first defined from the applicable metrics; controlled paraphrases, incomplete criteria, and unsupported criteria were then generated; and the corresponding gold outputs were instantiated from the formal model and manually reviewed. The complete experimental material is publicly available.[2]

All cases were independently executed through the OpenAI Responses API using `gpt-5.2-2025-12-11` with low reasoning effort, the same common prompt, view-specific module, examples, and strict JSON schema. All 420 executions returned schema-conformant JSON.

We compared the full configuration with two ablations. In *without retrieved quality metadata*, the quality catalog was removed, while view, ontology, materialization, and provenance metadata were preserved. The prompt allowed dimensions, metrics, levels, evidence types, units, and operators to be inferred rather than automatically returning `insufficient_context`. In *without retrieved view metadata*, the view specification, ontology, materialization, and provenance blocks were removed, while the target-view descriptor and quality metadata were preserved; semantic-scope elements could therefore be inferred.

All configurations used the same cases, gold outputs, model, modules, examples, and schema. The ablations consequently measure the incremental contribution of retrieved metadata rather than the complete absence of quality or view knowledge. Complete contexts were available only to the offline verifier.

5.2. Evaluation Measures

The evaluation reports five effectiveness measures and four operational indicators.

Status accuracy evaluates whether the agent returns the interpretation status defined in the gold standard: valid, clarification required, or one of the unsupported or rejected statuses. This measure is computed over all 140 test cases.

Core-requirement accuracy is computed over the 96 valid cases. A core requirement is considered correct when the generated output agrees with the gold standard on the target view, quality dimension, evaluation level, observed metric, required semantic scope, comparison operator, and threshold. This measure captures the elements needed to operationalize the quality condition directly expressed by the user criterion, independently of whether an expected metric is also selected.

Expected-metric selection is evaluated using *coverage* and *precision*. *Expected-metric coverage* is the proportion of valid cases containing an expected metric in the gold standard for which the agent returns that metric correctly. *Expected-metric precision* is the proportion of expected metrics returned by the agent that match the expected metric in the gold standard. Coverage, therefore, evaluates how frequently the agent identifies an available specification-level interpretation, whereas precision evaluates whether the metric is correct when the agent decides to include it.

Grounding validity is the proportion of requirements classified as valid whose controlled resources and values are present in the complete reference metadata for the target view. The verification includes classes, properties, predicates, functions, dimensions, metrics, evaluation levels, evidence types, units, and operators. The complete reference context is used only by the offline verifier and is not supplied to the model in the ablation configurations.

Finally, the operational comparison reports the average number of input tokens, total tokens, reasoning tokens, and wall-clock execution time over the 140 executions of each configuration.

[2]Available at https://tiny.cc/rq-agent-experiment.

Table 2
Effectiveness results for the full configuration and metadata ablations.

Configuration / view	Status	Core	Expected coverage	Expected precision	Grounding
Full – Exported	33/35 (94.3%)	22/24 (91.7%)	13/24 (54.2%)	13/13 (100%)	26/26 (100%)
Full – Linkset	35/35 (100%)	21/24 (87.5%)	18/24 (75.0%)	18/18 (100%)	24/24 (100%)
Full – Unification	35/35 (100%)	23/24 (95.8%)	8/20 (40.0%)	8/8 (100%)	24/24 (100%)
Full – Fusion	35/35 (100%)	21/24 (87.5%)	7/20 (35.0%)	7/7 (100%)	24/24 (100%)
Full – Overall	**138/140 (98.6%)**	**87/96 (90.6%)**	**46/88 (52.3%)**	**46/46 (100%)**	**98/98 (100%)**
Without quality metadata	136/140 (97.1%)	26/96 (27.1%)	49/88 (55.7%)	49/56 (87.5%)	3/100 (3.0%)
Without view metadata	130/140 (92.9%)	51/96 (53.1%)	46/88 (52.3%)	46/47 (97.9%)	67/86 (77.9%)

5.3. Results and Discussion

Table 2 presents the effectiveness results. Status accuracy is computed over all criteria, whereas core-requirement accuracy is computed over the 96 valid criteria. Expected-metric coverage uses the 88 valid cases for which the gold standard defines an expected metric, and expected-metric precision is computed over the cases in which the agent actually returned one. Grounding validity is computed over the requirements classified as valid by each configuration.

The full metadata-grounded configuration correctly classified 138 of the 140 criteria, corresponding to 98.6% status accuracy. Core-requirement accuracy reached 90.6%, with 87 of the 96 valid criteria correctly formalized. Expected-metric coverage reached 52.3%, and all 46 selected expected metrics were correct, resulting in 100% precision. All 98 requirements classified as valid used only resources and controlled values available in the reference metadata.

A closer analysis of the expected-metric results reveals an asymmetry between the two metric roles. Observed metrics characterize quality evidence evaluated over the materialized data, whereas expected metrics capture quality expectations associated with the semantic-view specification. Among the 42 cases in which an expected metric defined in the gold standard was not recovered, the observed metric was nevertheless correctly identified in 40 cases. This suggests that the main difficulty lies in identifying the specification-level quality perspective rather than the data-level quality aspect. At the same time, whenever the agent returned an expected metric, the selection was correct. Thus, the current configuration favors reliable expected-metric selection at the cost of lower coverage.

Removing the retrieved quality metadata had little effect on status classification, which decreased from 98.6% to 97.1%. However, core-requirement accuracy decreased sharply from 90.6% to 27.1%. The model could still recognize whether a criterion appeared to express a quality requirement, but it could not reliably reconstruct the controlled quality dimension, evaluation level, evidence types, operators, units, and metric applicability without the retrieved quality catalog.

Expected-metric coverage increased slightly to 55.7% in this ablation, but precision decreased from 100% to 87.5%. This increase in coverage should therefore not be interpreted as an improvement: the agent selected expected metrics more frequently but introduced seven incorrect selections. This trade-off suggests that quality metadata helps constrain unsupported specification-level metric selections, although the relatively low coverage in the full configuration also indicates that the interpretation strategy can still be improved. More importantly, only three of the 100 requirements classified as valid used exclusively resources available in the reference metadata, resulting in 3.0% grounding validity. The outputs remained linguistically plausible, but most were not grounded in the controlled DQA vocabulary of the evaluated EKG.

Removing the retrieved view metadata produced a different error pattern. Status accuracy decreased to 92.9%, and core-requirement accuracy decreased to 53.1%. Expected-metric coverage remained unchanged at 52.3%, while precision decreased only slightly to 97.9%. This pattern suggests that expected-metric selection is more sensitive to quality metadata than to view metadata, while the latter primarily contributes to grounding the semantic scope.

Grounding validity without view metadata reached 77.9%, substantially higher than in the quality-metadata ablation but lower than in the full configuration. The errors were concentrated in semantic-scope elements such as source and target classes, fused-entity types, node kinds, domains, and ranges.

The Linkset View was the most affected, with core accuracy decreasing to 16.7%, whereas the Unification View retained 91.7%. The latter result reflects the fact that the unchanged static modules and examples already contain several resources used by the Unification cases.

Both reductions in core-requirement accuracy were significant under McNemar's exact test ($p < 0.001$). The status difference was significant only without view metadata ($p = 0.039$), but not without quality metadata ($p = 0.625$). The results indicate complementary roles: quality metadata constrains DQA concepts and applicability, whereas view metadata grounds the semantic scope.

The full, no-quality-metadata, and no-view-metadata configurations used mean input-token counts of 8,860, 5,315, and 6,811, respectively, corresponding to reductions of 40.0% and 23.1% for the two ablations. Mean total-token counts decreased by 36.6% and 21.0%, respectively. However, reasoning-token usage increased from 133 in the full configuration to 185 (38.5% more) without quality metadata and 158 (18.8% more) without view metadata. The observed mean execution times were 8.72, 9.18, and 9.11 seconds, respectively. Thus, in these runs, the smaller contexts reduced token volume but did not reduce mean execution time and instead required more reasoning tokens.

The experiment constitutes a controlled feasibility evaluation based on synthetically generated criteria and a manually reviewed synthetic gold standard. Although this design enables controlled coverage of view types, metrics, paraphrases, ambiguities, and negative cases, it is limited to one music-domain EKG, one language model, and one execution per criterion. Because the static modules and examples were preserved, the ablations measure the incremental contribution of dynamically retrieved metadata rather than the complete absence of view or quality knowledge. The evaluation does not include a non-LLM baseline and therefore does not establish superiority over simpler formalization strategies. The approach also assumes sufficiently rich view and quality metadata in the target EKG, and its applicability when such metadata is unavailable remains to be evaluated. Finally, the current model focuses on threshold-based requirements; more complex conditional, temporal, multi-metric, and provenance-dependent requirements remain outside the present evaluation.

Future evaluations should include domain-expert requirements, additional EKGs and domains, repeated executions, a prompt-only configuration that removes both retrieved metadata components, and prompting strategies that assess observed and expected quality perspectives more explicitly and independently.

6. Related Work

A substantial body of work has focused on translating natural-language requirements into formal artifacts. Req2Spec [5] is an early example that transforms software requirements into formal specifications using Natural Language Processing (NLP) techniques. More recently, Large Language Models (LLMs) have been used to achieve automated conversion of Natural Language Requirements (NLRs) to Computation Tree Logic (CTL) [6] and Linear Temporal Logic (LTL) [7] to support the formal verification of software systems. Although these approaches share our goal of bridging Natural Language (NL) and formal semantics, they target program verification rather than data quality, and their output is not represented in RDF. Closer to our representational choices is the work of Westermann et al. [8], which employs an LLM to translate textual engineering rules into SHACL constraints validated against an AutomationML ontology, with the validation results subsequently explained back in NL — a pipeline structurally similar to ours (NL $\longrightarrow$ RDF constraint $\longrightarrow$ validation). While this approach demonstrates the feasibility of generating RDF constraints from natural-language text, it does not model the elements formalized by our approach—namely, quality dimension, selected metrics, semantic scope, operator, threshold, and evaluation level—nor does it rely on a dedicated quality metadata layer to represent metric applicability and evidence requirements.

Specifically, turning to the Data Quality (DQ) domain, researchers have explored various rule-based and ontology-driven frameworks for specifying user requirements. For example, González et al. [9] propose BIGOWL4DQ, an ontology to describe business data quality rules, characteristics, measurements, and assessments that reuses DQ-Vocab concepts and incorporates DMN4DQ, a rule-based

mechanism to represent user requirements on data quality within a given context of use. In contrast to our approach, however, requirements in DMN4DQ are expressed as pre-defined rules rather than derived from natural-language criteria, and the framework does not specifically target EKG semantic views. Ying and Gang [10] address Knowledge Graph (KG) completeness assessment guided by user requirements; nevertheless, their approach is restricted to a single quality dimension (completeness), does not rely on a dedicated metadata graph to disambiguate the requirement, and does not produce a reusable RDF-based representation of the resulting requirement.

Beyond the specification and representation of requirements, a parallel strand of research concerns the use of LLMs to translate natural-language questions into executable SPARQL queries over KGs. Emonet et al. [11] propose a Retrieval-Augmented Generation (RAG) system that leverages KG metadata — including schema information and query examples — to improve the accuracy of federated SPARQL query generation. This approach is conceptually close to ours in its use of metadata as disambiguation context. FIRESPARQL [12] and SPARQL-LLM [13] similarly report high accuracy in domain-specific Knowledge Graph Question Answering (KGQA) settings. These systems, however, solve a different problem: they translate a question into an ad-hoc retrieval query intended for immediate execution, whereas our agent produces a formal, reusable quality requirement from which multiple DQA mechanisms — including, but not limited to, SPARQL queries — can subsequently be instantiated.

Overall, existing approaches address natural-language formalization, RDF constraints, data-quality requirements, and executable query generation largely as separate concerns. Our approach combines these perspectives for EKG semantic views by grounding user-defined quality criteria in view and quality metadata, and by producing a reusable formal RDF requirement that is independent of any specific downstream execution mechanism.

7. Conclusion

This paper presented and evaluated a metadata-driven Requirement Interpretation Agent for formalizing natural-language quality criteria over EKG semantic views. The evaluation focuses on the interpretation and formalization stage of the broader requirement-driven DQA workflow. The full configuration achieved 90.6% core-requirement accuracy and 100% grounding validity, while expected metrics achieved 100% precision at 52.3% coverage. Ablation results indicate complementary roles for the retrieved metadata: quality metadata constrains DQA concepts and applicability, whereas view metadata grounds the semantic scope. These results support the feasibility of the approach in the evaluated setting but do not yet establish broad generalization.

Future work will evaluate expert-defined requirements, additional domains and EKGs, alternative models, baselines and prompting strategies, and the use of formalized requirements in downstream quality computation, assessment, diagnosis, and recommendation.

Acknowledgements

Narciso Arruda Jr. acknowledges the support of the Graduate Program in Computer Science (MDCC) at the Federal University of Ceará (UFC) for his participation in DMKG 2026. This work was partially supported by CNPq under Grant No. 306113/2026-0. The authors also acknowledge TSA Lynx Process (LYNX Otimização de Processos S.A.), Escola Superior Náutica Infante Dom Henrique, and Ci2 – Centro de Investigação em Cidades Inteligentes for their institutional support.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (OpenAI) and Grammarly to: check grammar and spelling, paraphrase and reword, and improve writing style. After using these tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey, Semantic Web 7 (2016) 63–93.

[2] J. Debattista, S. Auer, C. Lange, Luzzu: A framework for linked data quality assessment, in: 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), 2016, pp. 124–131.

[3] V. M. Vidal, M. A. Casanova, N. Arruda, M. Roberval, L. P. Leme, G. R. Lopes, C. Renso, Specification and incremental maintenance of linked data mashup views, in: International Conference on Advanced Information Systems Engineering, Springer, 2015, pp. 214–229.

[4] V. M. Vidal, R. Freitas, N. Arruda, M. A. Casanova, C. Renso, A data design pattern for building and exploring semantic views of enterprise knowledge graphs, in: Simpósio Brasileiro de Banco de Dados (SBBD), SBC, 2024, pp. 1–13.

[5] A. Nayak, H. P. Timmapathini, V. Murali, K. Ponnalagu, V. G. Venkoparao, A. Post, Req2spec: Transforming software requirements into formal specifications using natural language processing, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2022, pp. 87–95.

[6] M. Zhao, R. Tao, Y. Huang, J. Shi, S. Qin, Y. Yang, NL2CTL: Automatic generation of formal requirements specifications via large language models, in: K. Ogata, D. Mery, M. Sun, S. Liu (Eds.), Formal Methods and Software Engineering, Springer Nature Singapore, Singapore, 2024, pp. 1–17.

[7] Z. Ma, C. Wen, Z. Su, X. Liang, C. Tian, S. Qin, M. Yang, Bridging natural language and formal specification–automated translation of software requirements to ltl via hierarchical semantics decomposition using llms, arXiv preprint arXiv:2512.17334 (2025).

[8] T. Westermann, A. Köcher, F. Gehlhoff, Automated validation of textual constraints against automationml via llms and shacl, in: 2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA), 2025, pp. 1–4. doi:10.1109/ETFA65518.2025.11205616.

[9] C. Barba-González, I. Caballero, Ángel Jesús Varela-Vaca, J. A. Cruz-Lemus, M. T. Gómez-López, I. Navas-Delgado, Bigowl4dq: Ontology-driven approach for big data quality meta-modelling, selection and reasoning, Information and Software Technology 167 (2024) 107378. URL: https://www.sciencedirect.com/science/article/pii/S0950584923002331. doi:https://doi.org/10.1016/j.infsof.2023.107378.

[10] Z. Ying, X. Gang, How to implement a knowledge graph completeness assessment with the guidance of user requirements, Journal of Systems Engineering and Electronics PP (2024) 1–10. doi:10.23919/JSEE.2024.000046.

[11] V. Emonet, J. Bolleman, S. Duvaud, T. M. de Farias, A. C. Sima, Llm-based sparql query generation from natural language over federated knowledge graphs, arXiv preprint arXiv:2410.06062 (2024).

[12] X. Pan, V. de Boer, J. van Ossenbruggen, FIRESPARQL: A llm-based framework for SPARQL query generation over scholarly knowledge graphs, CoRR abs/2508.10467 (2025). URL: https://doi.org/10.48550/arXiv.2508.10467. doi:10.48550/ARXIV.2508.10467.

[13] P. Smeros, V. Emonet, R. Wang, A. C. Sima, T. M. de Farias, SPARQL-LLM: real-time SPARQL query generation from natural language questions, CoRR abs/2512.14277 (2025). URL: https://doi.org/10.48550/arXiv.2512.14277. doi:10.48550/ARXIV.2512.14277.