

KG Workbench: From Modular Ontologies to Human-Reviewed Knowledge Graphs

A Human-in-the-Loop Tool for LLM-assisted Knowledge Graph Construction

Ole Katzenberger^{1,*}, Ferdinand Schreiber¹, Christos Vagkidis², Alf Köhn-Seemann² and Thomas Kopinski¹

¹Faculty of Engineering and Business Sciences, South Westphalia University of Applied Sciences, Meschede, Germany

²Institute of Interfacial Process Engineering and Plasma Technology, University of Stuttgart, Stuttgart, Germany

Abstract

Knowledge graph (KG) construction is an iterative process that requires reliable ontology design, controlled fact extraction, and careful review before new triples are integrated into a target graph. This is especially important in grounding-oriented AI settings such as GraphRAG, where answer quality depends on the traceability and semantic correctness of the underlying graph. In practice, however, ontology modeling, extraction, and review are often distributed across separate tools and workflows, making semantic decisions harder to inspect and revise.

This paper presents the KG Workbench, an interactive system that combines Modular Ontology Modeling (MOMo), ontology-guided fact extraction setup, and human-in-the-loop triple review in a single environment. Building on MOMo, the system supports the creation or import of modular ontologies through a visually assisted modeling interface, selection of ontology scope and language for schema-based, LLM-assisted extraction, and review of extracted facts through document-based, tabular, and graph-based views. Candidate facts can be accepted, rejected, corrected, or remapped before downstream integration. Using a fusion-domain scenario as an illustrative example, we show how the tool is intended to make KG construction more reviewable, controllable, and practically integrated across these steps.

Keywords

Knowledge Graph Construction, Modular Ontology Modeling (MOMo), Large Language Models, GraphRAG

1. Introduction

Knowledge graph (KG) construction combines ontology design, triple extraction, validation, and integration. Although LLMs can support these steps, they still require domain-specific control, especially when KGs serve as grounding layers for RAG and GraphRAG.

Modular Ontology Modeling (MOMo) supports this by structuring ontologies into coherent, revisable modules, improving maintainability and creating controlled contexts for KG engineering [1, 2]. However, textual modeling alone offers limited oversight, and automatic extraction may produce plausible but incorrect triples.

This paper presents an interactive web application that combines MOMo with ontology-guided triple review. It supports visual and tabular inspection, human correction, and potential points of connection for later graph integration, with the aim of enabling more transparent and reliable KG construction.

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

*Corresponding author.

✉ katzenberger.ole@fh-swf.de (O. Katzenberger); schreiber.ferdinandmoritz@fh-swf.de (F. Schreiber); christos.vagkidis@igvp.uni-stuttgart.de (C. Vagkidis); alf.koehn@igvp.uni-stuttgart.de (A. Köhn-Seemann); kopinski.thomas@fh-swf.de (T. Kopinski)

🌐 https://www.fh-swf.de/de/ueber_uns/standorte_4/meschede_4/fb_iw/dozentinnen/mitarbeiter_6/katzenberger/index.php (O. Katzenberger);

https://www.fh-swf.de/de/ueber_uns/standorte_4/meschede_4/fb_iw/dozentinnen/mitarbeiter_6/schreiber/index.php (F. Schreiber);

<https://www.igvp.uni-stuttgart.de/team/Vagkidis-M.Sc./> (C. Vagkidis);

<https://www.igvp.uni-stuttgart.de/team/Koehn-Seemann/> (A. Köhn-Seemann);

https://www.fh-swf.de/de/ueber_uns/standorte_4/meschede_4/fb_iw/dozentinnen/professorinnen_1/kopinski/index.php (T. Kopinski)

🆔 0009-0004-3201-6928 (O. Katzenberger); 0009-0002-0853-6873 (F. Schreiber); 0009-0000-8086-8843 (C. Vagkidis); 0000-0002-1192-2057 (A. Köhn-Seemann); 0009-0009-7440-6525 (T. Kopinski)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2. Modular Ontology Modeling

2.1. Purpose of Modularization

MOMo is based on the idea that an ontology should not be developed as one monolithic artifact, but as a set of semantically coherent modules [1, 2]. Each module represents a key notion of the domain and contains the classes, object and data properties, and axioms directly related to this notion. This modular structure is intended to reduce cognitive load for human experts, support focused discussions about specific parts of the domain, and make the ontology easier to understand, extend, and reuse.

In this sense, modular ontologies act as semantic bridges between expert conceptualization and formal data models [3, p. 21]. Schema diagrams strengthen this bridge by making domain structures visible and supporting review, especially in early modeling discussions where domain experts should not be required to work directly with formal OWL syntax [1, p. 473]. Modules can therefore be developed and discussed separately, while still being combined later into a coherent overall ontology.

2.2. The Workflow

MOMo follows an iterative workflow in which domain experts and ontology engineers jointly develop and refine the ontology. The process starts with a concrete use case, which defines why the ontology is needed and what kind of knowledge it should support. From this use case, competency questions (CQs) are formulated. These CQs describe what the ontology should be able to answer and therefore make the modeling requirements explicit. Based on the CQs, key terms and key notions are identified; these become the conceptual anchors around which modules are built [4, p. 6].

Where suitable patterns exist, MOMo selects and adapts them for the relevant key notions; otherwise, modules are modeled directly. The resulting modules are discussed visually, refined with domain experts, formalized in OWL, and finally integrated into a coherent ontology through cross-module links. Thus, the main effort of MOMo lies not only in producing a formal ontology file, but in building shared conceptual understanding between domain experts and ontology engineers [1, p. 475].

For this paper, MOMo provides the methodological basis, but ontology design patterns, detailed axiom engineering, and OWL reasoning are outside its current scope.

3. System Overview

3.1. Overall Workflow and System Boundary

The proposed system provides an integrated workspace for ontology design (purple area in Fig. 1), extraction setup (grey box in Fig. 1), and review of candidate facts (blue area in Fig. 1) before downstream use. Instead of distributing these tasks across separate environments, it keeps semantic decisions visible, revisable, and connected to later review steps. The overall workflow can be seen in Fig. 1: It starts either by creating an ontology in the tool or by importing an existing ontology or selected modules (node 3 in Fig. 2). The system persists the ontology structure itself as well as extracted facts, review states, and review decisions. By contrast, the extraction pipeline is not hardwired into the workbench, but invoked through an HTTP request to an external extractor service. The extractor endpoint can be defined through the environment configuration. The service receives the selected ontology, or only selected modules of it, together with the document from which facts should be extracted. The extractor may use any internal workflow or model, as long as it returns the extracted sections, entities, facts, confidence values, and evidence in the expected response structure. This boundary is intentional, as it allows users to test their own extraction workflows without modifying the workbench itself. In this paper, the extraction service is mocked and returns a hardcoded set of extracted facts; however, the authors plan to provide a default extractor implementation in future work.

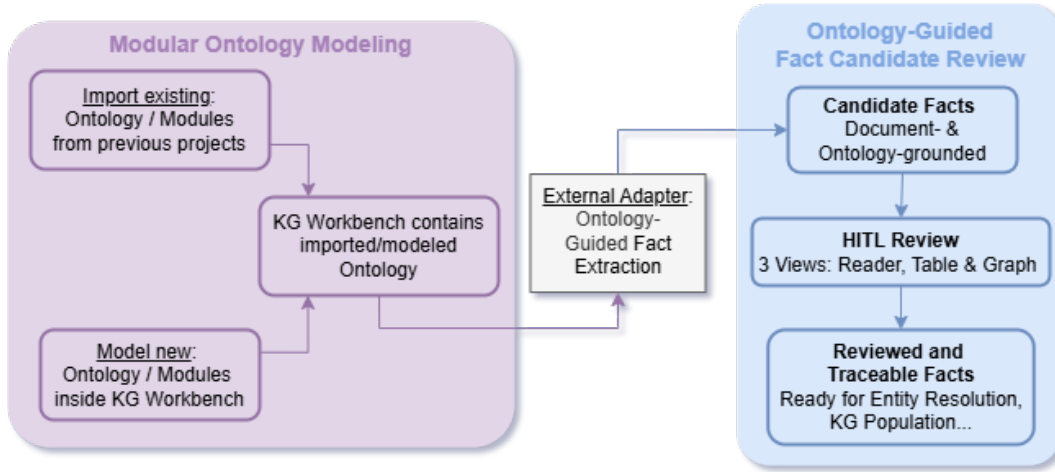


Figure 1: Workflow of the proposed system: modular ontology modeling or import is coupled with ontology-guided extraction and multi-view human-in-the-loop (HITL) review before reviewed facts may be prepared for KG integration.

3.2. Related Tools

KG Workbench is neither trying to replace Protégé’s¹ complete OWL engineering and reasoning capabilities nor to compete with the production graph infrastructure of Stardog², Graphwise³, or Neo4j⁴ - commercial or open-core platforms whose full functionality is not available as open source. Its contribution is that it closely follows MOMo, using modular ontology modeling as the foundation for ontology-guided fact extraction and evidence-linked human review, thereby connecting ontology work to trustworthy KG construction. Table 1 compares it with the mentioned tools.

Table 1

Comparison with selected ontology and knowledge graph tools. -/~✓ indicates no/partial/full support.

Tool	OWL authoring	Visual modeling	Modular modeling	Integration of CQs	Schema-guided extraction	HITL fact review
KG Workbench	~	✓	✓	✓	✓	✓
Protégé	✓	✓	~	-	-	-
Stardog	✓	✓	~	~	✓	~
Graphwise	✓	~	~	-	✓	~
Neo4j	~	✓	-	-	✓	-

3.3. Modeling the Ontology

The ontology authoring part of the system (node 1 in Fig. 2) translates the MOMo workflow introduced in Section 2.2 into an applicable tool. For this purpose, it provides the core elements required for ontology modeling, including CQs (visible by clicking node 4 in Fig. 2), modules (node 5 in Fig. 2), classes, relations, and class and relation properties (which support all RDF datatypes), translations, instances and notes (all beyond the mentioned nodes in Fig. 2).

¹<https://protegeproject.github.io/protege/>

²<https://docs.stardog.com/stardog-applications/>

³<https://help.graphwise.ai/?lang=en>

⁴<https://neo4j.com/developer/genai-ecosystem/importing-graph-from-unstructured-data/>

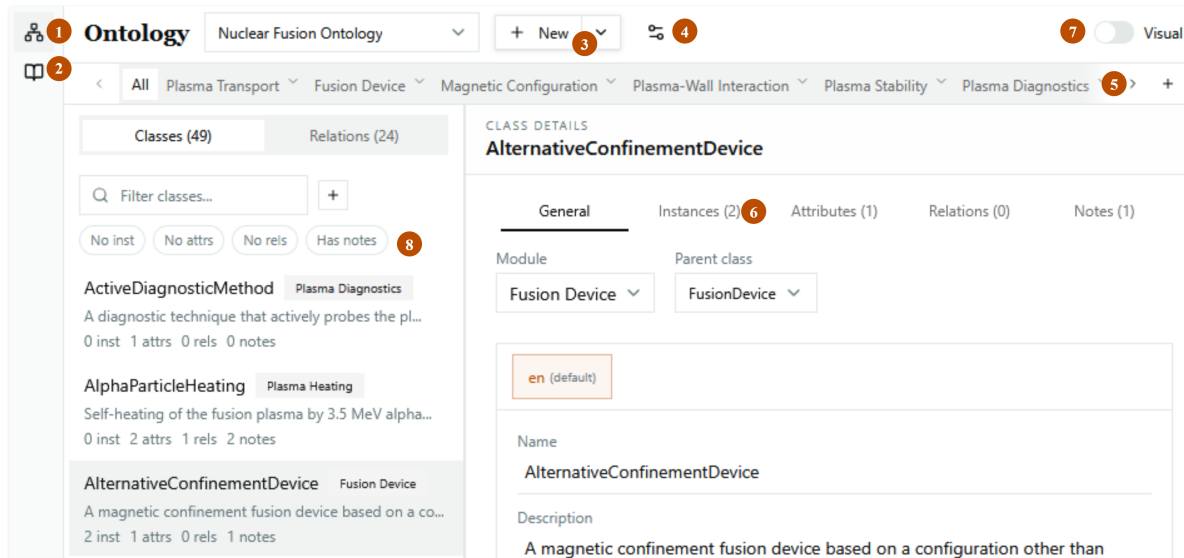


Figure 2: Ontology modeling interface with textual editing views for classes, instances, attributes, and relations, complemented by an optional graph mode for visual inspection of modules and cross-module relations.

3.3.1. Competency Questions and Instances as Modeling Anchors

Within the tool, CQs serve as explicit modeling anchors that can be linked to modules and thereby help relate formal ontology elements back to domain requirements. This is important because it keeps the connection between conceptual motivation and formal representation visible during later stages of the workflow. Since CQs can also be associated with subject, predicate, and object patterns as well as example instances, they additionally provide a direct bridge to the review workflow discussed in Section 3.5, where they can be used to filter the extracted graph and inspect whether the modeled questions can in fact be answered. The selectable instances are drawn from examples and instance candidates defined for classes and relations (node 6 in Fig. 2). Together with open or controlled instance policies, these examples make ontology elements more concrete for human experts and later provide guidance for extraction.

3.3.2. Visualization and Collaboration

One of the main concerns that motivated the development of the tool is the need for visual inspection: after activating visual mode (node 7 in Fig. 2), modules can be viewed together with their internal structure and cross-module relations, which may support discussion without requiring all participants to work directly with OWL syntax. Lightweight notes complement this by providing a simple mechanism for documenting modeling decisions, open questions, and review comments during ontology development. In addition, classes and relations can be filtered according to whether they already have or still lack properties, relations, or instances, and whether notes are attached to them (node 8 in Fig. 2). This makes it easier to identify parts of the ontology that still need enrichment or further discussion.

Finally, the workflow is designed to remain portable and iterative: Ontology content can be exported as JSON or OWL, with selectable module scope and language, and imported again for further refinement (all in node 3 in Fig. 2).

3.4. Ontology-Guided Extraction Setup

One of the central contributions of the system is that ontology modeling and extraction are coupled directly in the extraction setup. Before an extraction run is started, the user uploads the document to be processed and selects the ontology, language, and module scope that should guide the extraction. This

narrows the semantic search space and supports more specific extraction. Since module selection is dependency-aware, cross-module class and relation dependencies can be resolved before extraction if needed.

The selected ontology structure provides schema information for generating candidate facts according to the expected classes, relations, properties, and constraints. The actual extraction is handled by an external adapter and is outside the scope of this paper, allowing different backends to be connected as long as they follow the expected API response contract.

3.5. Human-in-the-Loop Review and Correction

After extraction, the system provides the HITL review workspace (reachable via node 2 in Fig. 2) that can be seen in Fig. 3, in which candidate facts can be inspected in the source document, in a graph visualization and in a table (node 1 in Fig. 3). This connects extracted triples with their textual evidence and surrounding graph context, making it clear which source passage supports each fact.

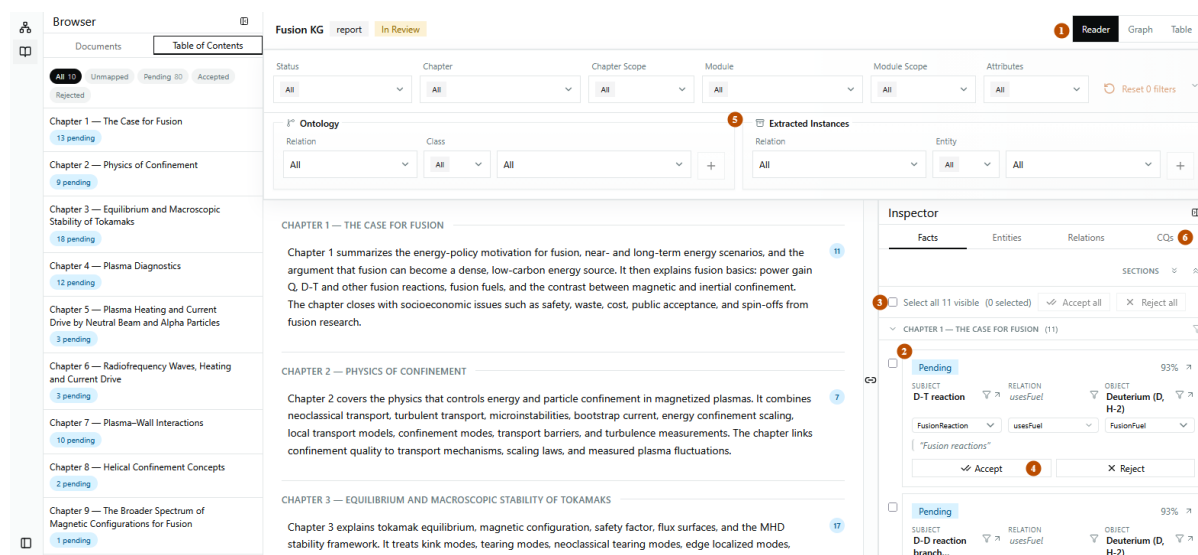


Figure 3: Review and correction workspace: users can inspect candidate facts in the source document, table, and graph visualization.

3.5.1. Review and Correction Workflow

Users can accept, reject, or correct candidate facts by renaming text spans, remapping relation types, changing entity classes, and editing properties (node 2 in Fig. 3). The review step therefore serves not only as validation, but also as semantic refinement before downstream graph integration. Bulk acceptance and rejection are supported (node 3 in Fig. 3), which may be especially useful in the table view after pre-filtering result sets according to specific review criteria.

Review is guided by the selected ontology. Candidate facts are checked against ontology classes, relations, and required properties, so acceptance depends on semantic completeness as well as surface plausibility. For example, facts can only be accepted once required properties are filled and subject and object entities satisfy the constraints of their assigned classes (node 4 in Fig. 3).

To preserve consistency, invalid remappings are not kept silently. Instead, the system surfaces their consequences and adjusts dependent mappings where necessary, helping to intercept plausible but misleading triples before graph integration.

3.5.2. Multi-Perspective Navigation and Filtering

To navigate larger result sets, shared filters apply across all three perspectives, including review status, document structure, module scope, classes, entities, and relations (node 5 in Fig. 3). Filters can be

triggered directly from filterable elements or combined for more targeted inspection. As mentioned in Section 3.3.1, CQs can be used as one-click filters as well (node 6 in Fig. 3). This allows users to filter the extracted graph directly by modeled questions and assess whether the reviewed facts are sufficient to answer them.

4. Discussion and Outlook

The main contribution of the presented tool lies in integrating MOMo, ontology-guided extraction setup, and evidence-linked HITL review in one workflow. Instead of treating MOMo as a purely conceptual activity, the tool operationalizes modular ontology modeling as a basis for traceable KG construction: ontology scope, language, and module context can guide extraction, while human experts can inspect, correct, and contextualize candidate facts before downstream graph integration.

The proposed approach does not yet benchmark extraction quality or review efficiency, and the interchangeable extraction through an external HTTP service is currently demonstrated with a hard-coded mock response. It also does not replace full OWL authoring or reasoning environments. Future work should therefore focus on integrating and comparing interchangeable extraction pipelines, adding transparent entity resolution, and turning accepted facts into a direct, user-controlled graph import process. Further extensions could use review outcomes as feedback for ontology refinement and extraction configuration, and selectively support additional MOMo elements such as pattern-informed modeling and basic axiom engineering without shifting the tool toward full OWL reasoning. Once these remaining workflow components are implemented, usability tests should be conducted to evaluate the tool's practical usefulness for domain experts.

Acknowledgments

The authors would like to thank FuseNet⁵ for their valuable feedback and support in the validation and refinement of the ontology presented in this paper.

Declaration on Generative AI

During the preparation of this work, the authors used Codex with GPT-5.4 and Claude with Sonnet 4.6 for code generation and refactoring during tool implementation. In addition, Codex with GPT-5.4 was used in the development of the use case ontology within a MOMo-as-Wiki repository and for support in editing the paper, including improving grammar and refining formulations. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] C. Shimizu, K. Hammar, P. Hitzler, Modular ontology modeling, *Semantic Web* 14 (2023) 459–489. doi:10.3233/SW-222886.
- [2] C. Shimizu, P. Hitzler, Accelerating knowledge graph and ontology engineering with large language models, 2024. URL: <http://arxiv.org/abs/2411.09601>. arXiv: 2411.09601, accessed: June 29, 2025.
- [3] J. Barrasa, A. E. Hodler, J. Webber, *Knowledge Graphs*, 1 ed., O'Reilly Media, Sebastopol, CA, 2021.
- [4] P. Hitzler, A. A. Krisnadhi, A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology, 2018. URL: <http://arxiv.org/abs/1808.08433>. arXiv: 1808.08433, accessed: April 17, 2025.

⁵<https://fusenet.eu/>