



Electronic Communications of the EASST
Volume 86 (1) Year 2026

deRSE26 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany (Part 1)

Edited by Bernd Flemisch, René Caspart, Bernadette Fritzsch, Florian Goth, Jean-Noël Grad, Florian Thiery

AI-Assisted RDF Authoring from JSON Data

Felix Neubauer, Mahdi Jafarkhani, Kenichi Endo, Jürgen Pleiss, Benjamin Uekermann

DOI: 10.14279/eceasst.v86.1.2789

License: This article is licensed under a CC-BY 4.0 License.

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

AI-Assisted RDF Authoring from JSON Data

Felix Neubauer [1], **Mahdi Jafarkhani**[1], **Kenichi Endo** [2], **Jürgen Pleiss** [3], **Benjamin Uekermann** [1]

[1] Institute for Parallel and Distributed Systems, University of Stuttgart, [2] Institute of Polymer Chemistry, University of Stuttgart, [3] Institute of Biochemistry, University of Stuttgart
felix.neubauer@ipvs.uni-stuttgart.de

Abstract: Scientific workflows increasingly produce structured JSON data that is easy to exchange but hard to interpret consistently across systems: JSON Schema validates structure, but carries no Linked Data semantics. We present an *RDF Authoring View* that extends the open-source JSON Schema editor MetaConfigurator into an integrated, AI-assisted environment for turning existing JSON, YAML, XML, or CSV data into RDF within a single web interface. Researchers generate RML mappings with AI assistance, refine the resulting triples, edit JSON-LD `@context` definitions with ontology-aware IRI auto-completion, run SPARQL queries drafted from natural-language hints, and explore the data as an interactive knowledge graph. Across these steps, the AI drafts artifacts while the researcher stays in control through an explicit human-in-the-loop review. We demonstrate the workflow on laboratory data from metal-organic framework (MOF) synthesis, reuse established vocabularies aligned with the NFDI4Chem ecosystem, and validate the generated RDF against a SHACL shape.

Keywords: Semantic Web, RDF Authoring, JSON-LD, RML, Knowledge Graph, Visualization, SPARQL, AI-Assisted, Data Modeling, MetaConfigurator

1 Introduction

The Semantic Web and its Resource Description Framework (RDF) [MM04] aim to improve interoperability, make semantics explicit, and enable cross-dataset integration [BHL01, JHHH15, HBC+21]. In practice, however, their adoption is hindered by ontology design effort, identifier and vocabulary management difficulties, and the complexity of mapping heterogeneous data into RDF [JHHH15, HBC+21]. Computational experiments and simulation workflows, for example, produce large volumes of scientific data [CBC+17, WAB+25, VKM+25] that could, in principle, be modeled and validated directly with RDF-based standards [CWL14, W3C17], yet researchers still often work with spreadsheets or machine-readable formats such as XML and JSON rather than native RDF graphs [Pat19, Kou19, BCG20, KJP+20].

Declarative transformation languages such as RDF Mapping Language (RML) make it possible to convert JSON, XML, or CSV data into RDF [DVC+14]. However, defining mappings, editing RDF triples, querying resulting graphs, and visualizing them typically requires switching between several separate tools, and many existing systems focus on only one part of this workflow [CBI, HTN+14, USC20, Pro, CG, RDF, Red].

To address this gap, we extend MetaConfigurator[1], an open-source web application for schema-driven structured data editing [NBX+24], with an *RDF Authoring View*. MetaConfigurator is primarily a JSON Schema editor, which can also generate a web form for data instance editing, source code in different programming languages, and documentation based on a schema. Furthermore, it supports importing JSON, YAML, XML, and CSV data and inferring a schema from them [NPU25]. Regardless of the input serialization, MetaConfigurator represents imported data internally as JSON. Our extension integrates RML-based transformation to RDF, direct RDF and JSON-LD `@context` editing, ontology-aware Internationalized Resource Identifier (IRI) auto-completion, SPARQL querying, interactive knowledge graph visualization, and AI-assisted drafting of SPARQL queries and RML mappings from user hints. In this way, MetaConfigurator evolves from a schema-based data editor into an integrated environment for creating, transforming, querying, and exploring semantically enriched data.

This paper is based on the master's thesis of Jafarkhani [Jaf26], and documents the full implementation of the domain-agnostic workflow, demonstrated here with the metal–organic framework (MOF) synthesis data of Neubauer et al. [NEB+26] as a chemistry example. We give a condensed account of its AI-assisted, human-in-the-loop workflow and of making the resulting RDF interoperable within the NFDI4Chem ecosystem through established-vocabulary reuse and SHACL validation. We first introduce the necessary background and the MOF synthesis use case (**Section 2**), then present the AI-assisted RDF authoring workflow applied throughout to that use case (**Section 3**), discuss its broader implications and positioning (**Section 4**), and conclude with its limitations and outlook (**Section 5**).

2 Background and the MOF Synthesis Use Case

RDF represents information as subject–predicate–object triples whose terms are IRIs drawn from shared vocabularies, so that the same statement can be interpreted consistently across datasets and tools [CWL14]. JSON-LD is a JSON serialization of RDF [SKL14]: a `@context` block maps the short keys of an ordinary JSON document to full IRIs, which is precisely what turns familiar JSON into Linked Data without changing its shape. SPARQL is the standard query language for RDF graphs [PS13].

As a running example in this paper we use laboratory data from the synthesis of metal–organic frameworks (MOFs) reported by Neubauer et al. [NEB+26]. MOFs are a class of porous materials with promising applications whose synthetic procedures involve many critical variables that are often difficult to document and analyze [CWB25, CTJ+26]. To make such procedures machine-readable, structured JSON formats have recently been proposed [NEB+26, CTJ+26, WAR+26]. This makes MOF synthesis a representative and demanding test case for our workflow: the data is deeply nested and heterogeneous (reagents, ordered procedure steps, and quantities with units), it is authored by chemists rather than RDF experts, and its value for reuse depends on capturing protocol-level detail with explicit semantics.

For the resulting RDF to be reusable within the German National Research Data Infrastructure (NFDI), it should reuse established vocabularies rather than mint terms ad hoc. We therefore draw on OBO Foundry ontologies [JMO+21] for the domain terms: ChEBI for chemical

[1] https://metaconfigurator.org

entities [HOD+16], and the Core Ontology for Biology and Biomedicine (COB) [OBO25], the Ontology for Biomedical Investigations (OBI) [BBB+16], the Basic Formal Ontology (BFO) [ASS15], and the Relation Ontology (RO) [SCK+05] for processes, steps, and their relations, together with QUDT for quantities and units [QUD25]. Several of these established ontologies, including ChEBI, OBI, BFO, and QUDT, are also reused by ChemDCAT-AP, the chemistry metadata profile developed by NFDI4Chem [SKB+20] and NFDI4Cat [WBB+21, SBL+26], so our output shares its semantic foundations, and all of the OBO and QUDT terms we use can be resolved through NFDI terminology services (TS4NFDI) [Bas26]. Our data is not itself a ChemDCAT-AP instance, though: it captures synthesis procedures at the protocol level, whereas ChemDCAT-AP describes datasets at the metadata level. We revisit this alignment, and validate the generated RDF against a SHACL shape, in Section 3.5.

3 AI-Assisted RDF Authoring

The *RDF Authoring View* lets users transform, inspect, edit, query, and visualize RDF directly within MetaConfigurator. Because RDF is represented internally as JSON-LD, the data must be in JSON-LD before the RDF-specific features become available; if it is not, the view offers to convert the currently loaded data. We walk through the workflow on the MOF synthesis data of **Listing 1**, which encodes one complete synthesis procedure identified as `S-1`. Two of its steps (generating the RML mapping and drafting SPARQL queries) are supported by AI assistance, and both keep the researcher in control through an explicit review before the result is applied or executed.

Minor adjustments were made to the source data to support the transformation: for the lists of steps, the ordering was captured explicitly in a new field, and the unit strings were aligned with QUDT units. The complete input and all derived artifacts are archived on DaRUS [Neu26].

3.1 From JSON to JSON-LD via RML

The first step converts the JSON into JSON-LD using an RML mapping. In the mapping dialog, users either paste an existing RML mapping or request a suggestion from the AI assistant by providing a natural-language hint (for example, which ontologies and terms to use). The suggested mapping is shown in a syntax-highlighted editor for inspection and refinement; a single action then executes it and produces the JSON-LD serialization. Creating a suitable mapping is the most demanding part of the workflow.

How the AI assistance works. The RML and SPARQL assistants follow the same hybrid pattern as MetaConfigurator's existing schema tooling [NUP25]: a prompt is assembled programmatically and sent to a configurable, OpenAI-compatible chat endpoint, and the textual response is post-processed before it reaches the user. The model is a user setting; we use `gpt-4o` for the minimal and failure cases and `gpt-5-mini` for the successful detailed mapping and SPARQL example. **Figure 1** shows the building blocks of the RML request. A fixed *system message* assigns the assistant its role and states strict RML/Turtle rules (use `rml:logicalSource`, construct absolute IRIs via `rr:template`, map only fields that exist, never invent values),

```
{
  "Synthesis": [
    {
      "Hardware": { "Component": [{ "_id": "S-1", "_type": "glass vial" }] },
      "Metadata": { "_description": "S-1", "_product": "MIL-88B (Fe)" },
      "Procedure": {
        "Prep": {
          "Step": [
            { "_vessel": "S-1", "$xml_type": "Add", "_amount": { "Unit": "MilliMOL", "Value": 0.4,
↪  "$xml_append": "${Value} ${Unit}" }, "_reagent": "FeCl3", "_order": 0 },
            { "_vessel": "S-1", "$xml_type": "Add", "_amount": { "Unit": "MilliMOL", "Value": 0.4,
↪  "$xml_append": "${Value} ${Unit}" }, "_reagent": "Benzene-1,4-dicarboxylic acid", "_order": 1 },
            { "_vessel": "S-1", "$xml_type": "Add", "_amount": { "Unit": "MilliL", "Value": 4, "$xml_append":
↪  "${Value} ${Unit}" }, "_reagent": "DMF", "_order": 2 }
          ]
        },
        "Reaction": {
          "Step": [
            { "_vessel": "S-1", "$xml_type": "Sonicate", "_time": { "Value": 30.0, "Unit": "MIN",
↪  "$xml_append": "${Value} ${Unit}" }, "_order": 0 },
            { "_comment": "In: oven", "_vessel": "S-1", "$xml_type": "HeatChill", "_temp": { "Unit": "DEG_C",
↪  "Value": 120.0, "$xml_append": "${Value} ${Unit}" }, "_time": { "Value": 1, "Unit":"DAY", "$xml_append":
↪  "${Value} ${Unit}" }, "_order": 1 }
          ]
        },
        "Workup": {
          "Step": [
            { "_vessel": "S-1", "$xml_type": "WashSolid", "_solvent": "EtOH", "_order": 0 },
            { "_vessel": "S-1", "$xml_type": "Dry", "_temp": { "Unit": "DEG_C", "Value": 120.0, "$xml_append":
↪  "${Value} ${Unit}" }, "_time": { "Value": 1, "Unit":"DAY", "$xml_append": "${Value} ${Unit}" },
↪  "_pressure": { "Unit": "PA", "Value": 0, "$xml_append": "${Value} ${Unit}" }, "_order": 1 }
          ]
        }
      },
      "Reagents": {
        "Reagent": [
          { "_id": "FeCl3", "_name": "FeCl3", "_role": "substrate" },
          { "_id": "Benzene-1%252C4-dicarboxylic%2520acid", "_name": "Benzene-1,4-dicarboxylic acid", "_role":
↪  "ligand" },
          { "_id": "DMF", "_name": "DMF", "_role": "solvent" },
          { "_id": "EtOH", "_name": "EtOH", "_role": "solvent" }
        ]
      }
    }
  ]
}
```


Listing 1: Representative JSON excerpt from the MOF synthesis data entry `S-1`

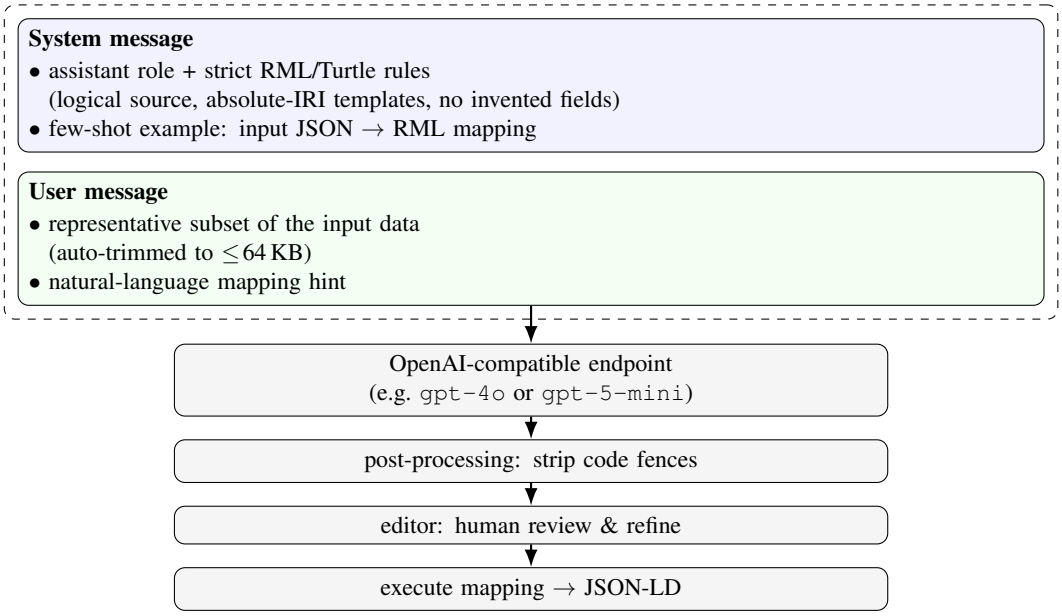


Figure 1: Anatomy of the AI request for RML mapping generation and the subsequent human-in-the-loop steps.

followed by a worked input/output example (few-shot). The *user message* carries a representative subset of the actual input data together with the user's hint. Because scientific data can be large, the subset is produced by a deterministic trimming algorithm that truncates arrays to n and object properties to $8n$ entries, halving n from 64 until the serialized subset fits within 64 KB (or n=2) [NUP25]; this keeps the prompt within context limits while preserving representative structure. The response is then stripped of Markdown code fences and shown in the editor. No mapping is applied automatically: the user reviews and edits the draft, and any structural errors surface deterministically when the mapping is executed.

The built-in assistant is optional: users may iterate with any external LLM service until the RML satisfies the intended graph structure, then paste the completed mapping into MetaConfigurator. The application executes it deterministically and provides the same visualization, editing, validation, and query facilities, without coupling the reproducible transformation to one LLM provider.

Mapping quality depends strongly on hint specificity. **Figure 2** contrasts results from a minimal and a detailed hint. Both panels show only the subgraph around the first preparation step of `S-1`; the detailed mapping covers the complete input.

A minimal hint. Given only *"Convert this JSON into RDF"*, the assistant returns a valid mapping that mirrors the JSON tree: it mints ad-hoc predicates and classes from the keys (`ex:hasProcedure`, `ex:hasPrepStep`, `ex:amount`), keeps amounts as opaque strings, and attaches reagents as literal names. The result (**Figure 2**a) is valid RDF but semantically no more interoperable than the original JSON.

```
Create a complete mapping without inventing terms from JSON keys. Model syntheses
and steps as planned processes, vessels as devices, and products and reagents as
chemical entities. Treat Prep, Reaction, and Workup separately.

[...]

The synthesis must be the subject of obo:BFO_0000051 and the step must be its
object. For each phase, add a relationship rule that reconstructs both IRIs from
the step record. Do not use parent-map references or cross joins.

[...]

Represent every amount, temperature, time, and pressure as a qudt:Quantity with
a separate qudt:QuantityValue. The step must be the subject of qudt:quantity and
the quantity its object. Only create optional quantities when their field exists.

[...]
```

Listing 2: Selected passages from the detailed natural-language mapping hint

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix obo: <http://purl.obolibrary.org/obo/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <https://schema.org/> .

<#SynthesisMap>
  rml:logicalSource [ rml:source "Data.json" ;
    rml:referenceFormulation ql:JSONPath ; rml:iterator "$.Synthesis[*]" ] ;
  rr:subjectMap [ rr:template "http://example.org/synthesis/{Hardware.Component[0]._id}" ; rr:class
  ↪  obo:COB_0000082 ] ;
  rr:predicateObjectMap [ rr:predicate schema:identifier ; rr:objectMap [ rml:reference
  ↪  "Hardware.Component[0]._id" ] ] ;
  rr:predicateObjectMap [ rr:predicate rdfs:label ; rr:objectMap [ rml:reference "Metadata._description" ] ] ;
  rr:predicateObjectMap [ rr:predicate obo:RO_0000057 ; rr:objectMap [ rr:template
  ↪  "http://example.org/vessel/{Hardware.Component[0]._id}" ; rr:termType rr:IRI ] ] ;
  rr:predicateObjectMap [ rr:predicate obo:OBI_0000299 ; rr:objectMap [ rr:template
  ↪  "http://example.org/product/{Metadata._product}" ; rr:termType rr:IRI ] ] .
[...]
```

Listing 3: Subset of the RML mapping focused on synthesis-level hardware and metadata

A detailed hint. A deliberately detailed natural-language hint specifies the target ontologies, IRI patterns, relationship directions, and handling of optional values (selected passages in **Listing 2**). We have a specific target graph rather than asking the model to invent one; the hint supplies domain knowledge but no RML/Turtle snippets. **Listing 3** shows an excerpt of the resulting mapping, and **Figure 2**b the RDF it produces: explicitly typed resources connected by shared predicates, with experimental parameters represented as QUDT quantities and values.

Vocabulary reuse. Following Section 2, synthesis runs and steps reuse the COB planned-process class, vessels are OBI devices, and products and reagents use ChEBI's general chemical-entity class (`obo:CHEBI_24431`). BFO, RO, and OBI connect these resources; QUDT represents parameters with numeric values and unit IRIs. Only instance IRIs use the local `ex:` namespace; reagents are not resolved to specific ChEBI classes.

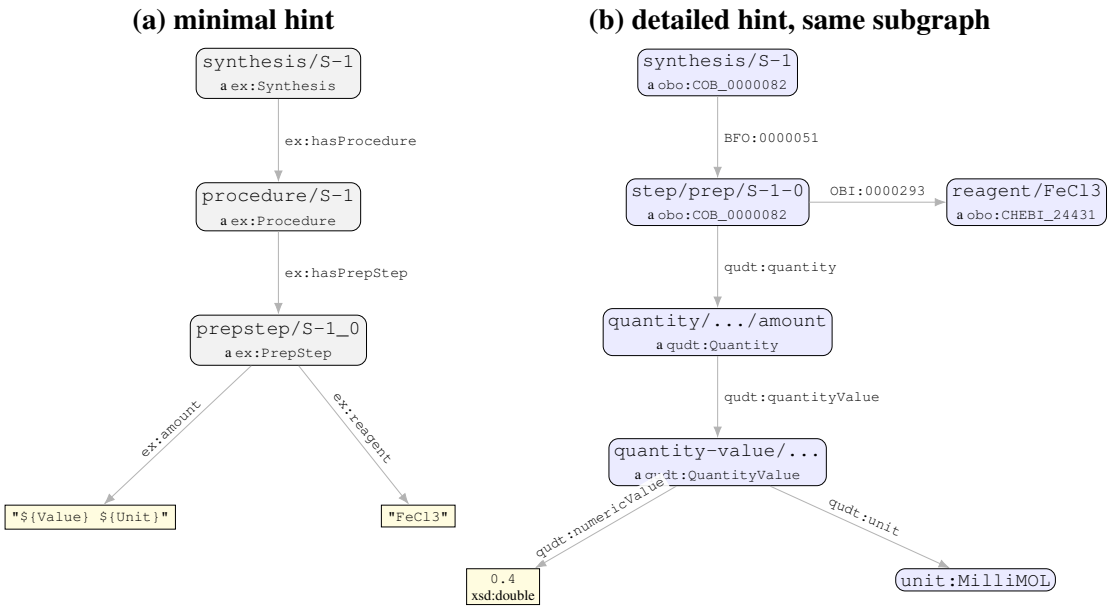


Figure 2: Matched subgraphs around the first preparation step of `S-1`, selected from the complete mapping results: (a) ad-hoc `ex:` terms and literal values from the minimal hint; (b) typed resources connected by shared OBO terms, with QUDT quantities, from the detailed hint.

3.2 Inspecting and Editing RDF

Once the data is in RDF, the *RDF Authoring View* supports inspection and editing through two tabs. Internally, the JSON-LD is parsed into an in-memory RDF store built on `rdflib.js` [lin26], whose statements back the table and editing actions. The first tab, *Triples* (**Figure 3**), lists these statements with paging, sorting, column and global filtering, and per-triple editing and export. Selection is synchronized bidirectionally with the JSON-LD text view: selecting a triple highlights the matching position in the document, and editing the document rebuilds the store. Users can also open the SPARQL dialog and graph visualization from this view.

The second tab, *Context*, edits the JSON-LD `@context`: the prefix-to-IRI mappings that ground the document's short keys (Section 2) and determine which ontologies the data refers to.

The context also drives ontology-aware editing. When a user edits a `NamedNode` predicate or object, the triple edit modal opens an *Ontology Explorer* (**Figure 4**) that extracts the declared ontology prefixes and offers their terms for auto-completion. Users can therefore select existing IRIs from a reused vocabulary instead of typing, mistyping, or inventing them by hand, which is where much of the interoperability of the resulting RDF is decided.

3.3 Querying with SPARQL and AI Assistance

The RDF graph can be queried with SPARQL in a dialog. Because `rdflib.js`'s own query support did not cover the required SPARQL 1.2 draft features, queries are executed with Comunica's in-browser query engine [Ass26b, Ass26a] against the in-memory store, with debounced live syntax validation via Sparql.js [Vc26]. As with RML, users can draft a query from a natural-

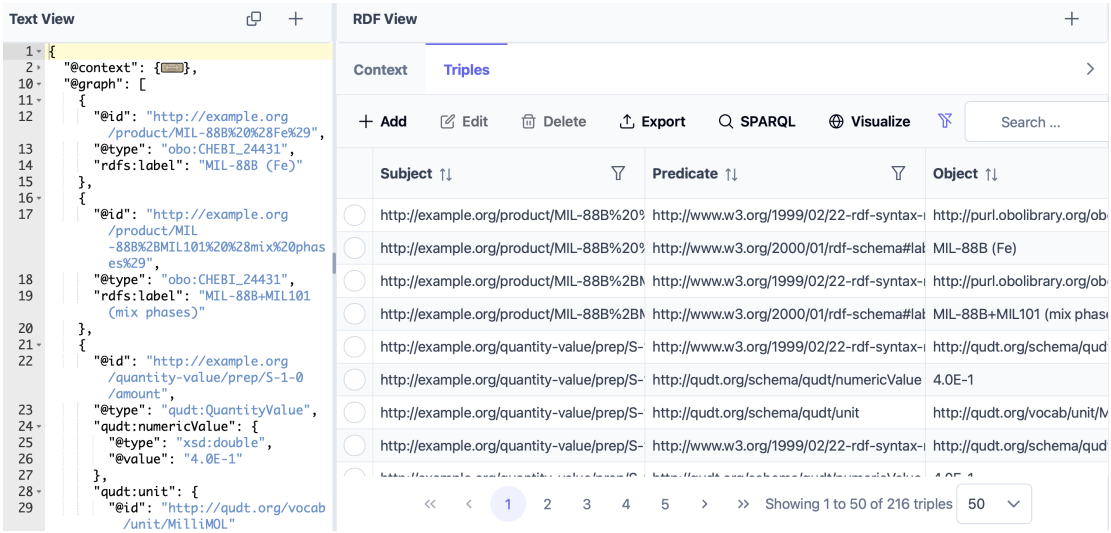


Figure 3: *Triples tab* in the *RDF Authoring View*.

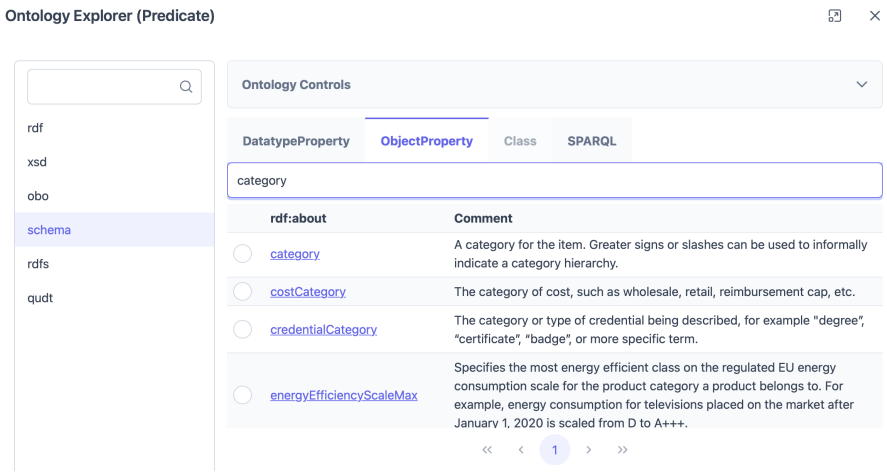


Figure 4: *Ontology Explorer dialog*.

language hint: the assistant assembles a prompt from fixed instructions, a `PREFIX` block derived from the current graph's namespaces, and a trimmed subset of the data, and returns a query for review; a visualization toggle switches the assistant between `SELECT` output and a graph-producing `CONSTRUCT` template. Writing correct SPARQL is a common bottleneck for domain experts, so this lets them start from an experimental question rather than from triple patterns. As with mappings, the generated query is a draft that may embed an imprecise predicate or modeling choice.

Figure 5 shows the full loop on the MOF graph: `gpt-5-mini` turns a natural-language hint into a SPARQL query, which is executed to retrieve, for each synthesis, its preparation steps together with the associated reagents and amounts.

3.4 Knowledge Graph Visualization

The same RDF can be explored as an interactive knowledge graph, rendered with Cytoscape.js [FLH+16] using a force-directed (COSE-Bilkent) layout. Resources appear as nodes and relations as labeled, directed edges, with namespace-aware prefix shortening for readability. The view supports zooming, fit-to-view, node search with type-ahead suggestions, image export, and clickable term links: a predicate label opens its ontology definition (e.g., `obo:BFO_0000051`), and an external object IRI such as `unit:MilliMOL` opens its QUDT definition, while an internal reference focuses the corresponding node. Selecting a node reveals its properties in a side panel and exposes the same authoring actions as the tabular views, so users can refine content directly from the graph. Large graphs are detected and confirmed before rendering. Because a graph can also be built from a SPARQL `CONSTRUCT` result, users can filter a large dataset semantically (e.g., only Prep-related resources) and visualize just the returned subgraph.

3.5 Where AI Assistance Helps, Where It Fails, and Validation

The AI assistance works well without intervention when the request maps closely to structures already present in the data and targets a small number of well-known terms: straightforward `SELECT` queries (listing entities of a type, or following a single named relation) and direct field-to-triple mappings are usually correct on the first attempt. It reaches its limits when the semantic intent is underspecified. With a vague hint the RML assistant produces valid but shallow mappings (Section 3.1). Even a fairly detailed hint can go wrong when it names the right ontologies but leaves the linking underspecified: in our runs, such a hint produced a mapping that joined parents to children without a join condition, so every synthesis was linked to the steps and reagents of *all* experiments (a Cartesian product) and the quantity nodes were dropped entirely. Model choice also mattered: with the same detailed natural-language prompt, `gpt-4o` reintroduced cross joins, whereas `gpt-5-mini` produced a graph that passed our structural checks. The SPARQL assistant similarly struggles with multi-hop patterns and aggregation. In every case the design keeps a human in the loop: drafts are shown rather than applied, checked syntactically (Sparql.js) and, for mappings, semantically at execution time.

Reusing shared vocabularies enables SHACL validation [W3C17]. Our shape (**Listing 4**) requires each synthesis with an outgoing has-part relation to have an identifier, label, and step, and each `qudt:QuantityValue` to have exactly one number and unit IRI. The 216 generated

Natural language hint

```
Retrieve all Prep steps with their reagents and amounts for each
↪  synthesis.
```

⇓

AI-generated SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX schema: <https://schema.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX qudt: <http://qudt.org/schema/qudt/>

SELECT ?synthesis ?synthesisId ?step ?stepLabel ?position ?reagent
       ?reagentLabel ?amount ?unit WHERE {
  ?synthesis obo:BFO_0000051 ?step ;
             schema:identifier ?synthesisId .
  ?step schema:category "Prep" ;
        rdfs:label ?stepLabel .
  OPTIONAL { ?step schema:position ?position . }
  OPTIONAL {
    ?step obo:OBI_0000293 ?reagent .
    ?reagent rdfs:label ?reagentLabel .
  }
  OPTIONAL {
    ?step qudt:quantity ?quantity .
    ?quantity qudt:quantityValue ?qv .
    ?qv qudt:numericValue ?amount .
    OPTIONAL { ?qv qudt:unit ?unit . }
  }
}
ORDER BY ?synthesis ?position
```

⇓

Query result (selected columns; IRIs shortened)

Synthesis	Step	Reagent	Value	Unit
S-1	prep/S-1-0	FeCl3	0.4	MilliMOL
S-1	prep/S-1-1	Benzene-1,4-dicarboxylic acid	0.4	MilliMOL
S-1	prep/S-1-2	DMF	4.0	MilliL
S-2	prep/S-2-0	FeCl3·6H2O	0.4	MilliMOL
S-2	prep/S-2-1	Benzene-1,4-dicarboxylic acid	0.4	MilliMOL
S-2	prep/S-2-2	DMF	4.0	MilliL

Figure 5: AI-assisted SPARQL workflow in the *RDF Authoring View*: a natural-language hint provided by the user is translated into a SPARQL query via AI-assistance, which is then executed deterministically to produce the query result.

```
@prefix sh:   <http://www.w3.org/ns/shacl#> .
@prefix obo:  <http://purl.obolibrary.org/obo/> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix schema: <https://schema.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix ex:   <http://example.org/> .

ex:SynthesisShape a sh:NodeShape ;
  sh:targetSubjectsOf obo:BFO_0000051 ;
  sh:class obo:COB_0000082 ;
  sh:property [ sh:path schema:identifier ; sh:minCount 1 ; sh:datatype xsd:string ] ;
  sh:property [ sh:path rdfs:label        ; sh:minCount 1 ] ;
  sh:property [ sh:path obo:BFO_0000051   ; sh:minCount 1 ; sh:nodeKind sh:IRI ] .

ex:QuantityShape a sh:NodeShape ;
  sh:targetClass qudt:QuantityValue ;
  sh:property [ sh:path qudt:unit         ; sh:minCount 1 ; sh:maxCount 1 ; sh:nodeKind sh:IRI ] ;
  sh:property [ sh:path qudt:numericValue ; sh:minCount 1 ; sh:maxCount 1 ; sh:datatype xsd:double ] .
```

Listing 4: SHACL shape used to validate the generated RDF

triples conform under `pySHACL` [SC26]. This shape can be extended toward the SHACL shapes shipped with ChemDCAT-AP.

4 Discussion

Semantic Web and Linked Data standards already provide a broad ecosystem for interoperable data, yet their adoption in research remains limited partly because creating mappings, selecting vocabularies, and querying RDF require specialist skills [JHHH15, HBC+21]. Research software can lower this barrier by making these capabilities accessible to domain experts who are not Semantic Web developers. MetaConfigurator follows this approach and uses AI in a deliberately constrained role: the model drafts code and other technical artifacts, which researchers can inspect and revise before they are executed deterministically. This combines the convenience of generative AI with transparent and reproducible processing [WAB+25]. The chemistry example illustrates how the domain-agnostic workflow fits into NFDI: it helps researchers apply shared vocabulary terms that are resolvable through TS4NFDI [BK23, Bas26] and produces RDF aligned with vocabularies also reused by NFDI4Chem. Applying the workflow in other fields would provide further evidence of its generalisability. The scope of MetaConfigurator can also be clarified in relation to existing tools. Karma and RDF Transform focus on defining RDF mappings for existing data [USC20, Ate26], whereas CEDAR and LinkML take template- and model-centered approaches to semantically described data [GOM+17, MSH+26]. MetaConfigurator combines schema-driven data authoring, RDF transformation and exploration, and reviewable AI drafting in one environment.

5 Conclusion

This paper presented the *RDF Authoring View*, an extension of MetaConfigurator that connects structured data with Semantic Web technologies in a single user interface through RML transformation, semantic editing, SPARQL querying, and interactive KG visualization. Its two AI-

assisted steps (drafting RML mappings and SPARQL queries) lower the entry barrier for non-expert users, while an explicit human-in-the-loop review before every apply or execute step keeps the user in control, and we discussed where this assistance is reliable and where it is not. The chemistry example based on MOF synthesis data shows that the generated RDF can reuse established OBO Foundry and QUDT vocabularies, several of them shared with the NFDI4Chem ChemDCAT-AP ecosystem, and validate against a SHACL shape.

At the same time, several limitations remain. AI-generated mappings and queries can still contain structural or semantic errors, so human review remains essential; for creating suitable RML mappings in particular, expert knowledge is indispensable. Scalability is limited by browser-side processing, especially for large graphs and interactive visualization. High-quality RDF authoring still depends on selecting appropriate ontologies, classes, predicates, and identifier patterns, which remains challenging for non-expert users, which is where vocabulary reuse, ontology-aware completion, and SHACL validation are meant to help.

Overall, the work demonstrates that user-centered semantic authoring can be embedded into an existing structured-data editor, providing a foundation for future Semantic Web tooling that is more accessible, more integrated, and better aligned with practical scientific workflows.

Author Contributions

Author contributions are provided using the CRediT (Contributor Roles Taxonomy) standardized contributor roles [BAA+15].

- Felix Neubauer: Conceptualization; Methodology; Validation; Supervision; Project administration; Writing – original draft; Writing – review & editing; Visualization.
- Mahdi Jafarkhani: Conceptualization; Methodology; Software; Validation; Investigation; Formal analysis; Writing – original draft; Visualization.
- Kenichi Endo: Writing – review & editing.
- Jürgen Pleiss: Supervision; Funding acquisition; Writing – review & editing.
- Benjamin Uekermann: Supervision; Funding acquisition; Writing – review & editing.

Acknowledgements: Financial support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant numbers 358283783 (SFB 1333/2 2022), 390740016 (EXC 2075), and 441958208 (NFDI4Chem) is gratefully acknowledged.

Data availability: The developed codes are publicly available on GitHub[2]. The complete application example, including input data, prompts, generated RML mappings and RDF results, SPARQL queries, and the SHACL shape, is archived on DaRUS [Neu26].

[2] https://github.com/MetaConfigurator/meta-configurator

Declaration on the use of AI: AI-based assistants (including OpenAI GPT and Claude) were used to support code development and limited editorial refinement. All generated outputs were critically reviewed, tested where applicable, and manually revised to ensure correctness and compliance with academic standards.

Bibliography

[ASS15] R. Arp, B. Smith, A. D. Spear. *Building Ontologies with Basic Formal Ontology*. MIT Press, 2015.
doi:10.7551/mitpress/9780262527811.001.0001

[Ass26a] C. Association. Comunica: Supported specifications. 2026. Accessed: 2026-07-06.
https://comunica.dev/docs/query/advanced/specifications/

[Ass26b] C. Association. @comunica/query-sparql. 2026. Accessed: 2026-07-06.
https://www.npmjs.com/package/@comunica/query-sparql

[Ate26] AtesComp. RDF Transform: An OpenRefine Extension for Transforming Data into RDF. 2026. Accessed: 2026-08-24.
https://github.com/AtesComp/rdf-transform

[BAA+15] A. Brand, L. Allen, M. Altman, M. Hlava, J. Scott. Beyond authorship: Attribution, contribution, collaboration, and credit. *Learned Publishing* 28(2):151–155, 2015.
doi:10.1087/20150211

[Bas26] Base4NFDI. TS4NFDI: Terminology Services for NFDI. 2026. Accessed: 2026-07-06.
https://base4nfdi.de/projects/ts4nfdi

[BBB+16] A. Bandrowski, R. Brinkman, M. Brochhausen, M. H. Brush, B. Bug, M. C. Chibucos, K. Clancy, M. Courtot, D. Derom, M. Dumontier, L. Fan, J. Fostel, G. Fragoso, A. Gonzalez-Beltran, M. A. Haendel, Y. He, M. Heiskanen, T. Hernandez-Boussard, M. Jensen, Y. Lin, A. Lister, J. Malone, E. Manduchi, M. McGee, J. A. Overton, H. Parkinson, B. Peters, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, D. Schober, B. Smith, L. N. Soldatova, C. J. Stoeckert, C. F. Taylor, C. Torniai, J. A. Turner, R. Vita, P. L. Whetzel, J. Zheng. The Ontology for Biomedical Investigations. *PLOS ONE* 11(4):e0154556, 2016.
doi:10.1371/journal.pone.0154556

[BCG20] K. A. Briney, H. Coates, A. Goben. Foundational Practices of Research Data Management. *Research Ideas and Outcomes*, 2020.
doi:10.3897/rio.6.e56508

[BHL01] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American* 284(5):34–43, 2001.
doi:10.1038/scientificamerican0501-34

[BK23] R. Baum, O. Koepler. Leveraging Terminology Services for FAIR Semantic Data Integration Across NFDI Domains: How to Integrate Terminology Services Into Other Service Applications. In *Proceedings of the Conference on Research Data Infrastructure*. Volume 1. 2023.
doi:10.52825/cordi.v1i.356

[CBC+17] S. Cohen-Boulakia, K. Belhajjame, O. Collin, J. Chopard, C. Froidevaux, A. Gaignard, K. Hinsen, P. Larmande, Y. Le Bras, F. Lemoine, F. Mareuil, H. Ménager, C. Pradal, C. Blanchet. Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems* 75:284–298, 2017.
doi:10.1016/j.future.2017.01.012

[CBI] S. CBIR. Protégé. Accessed: 2026-07-06.
https://protege.stanford.edu

[CG] S.-G. CG. SPARQL-Generate. Accessed: 2026-07-06.
https://github.com/SPARQL-Generate

[CTJ+26] O. Cheung, S. Tokuda, D. Jędrzejowski, E. Ploetz, B. Baumgartner, M. Pander, F. Yang, J. D. Evans, R. Ettlinger, S. Wuttke, D. Matoga. Material Preparation Information File (MPIF): A Community-Driven Standard for Reporting MOF Syntheses. *Advanced Materials* 38(16):e202521420, 2026.
doi:10.1002/adma.202521420

[CWB25] S. Canossa, S. Wuttke, L. J. Barbour. The Chemistry of Space: From Crystallographic Abstraction to Framework Design. *Angewandte Chemie International Edition* 64(52):e202523850, 2025.
doi:10.1002/anie.202523850

[CWL14] R. Cyganiak, D. Wood, M. Lanthaler. RDF 1.1 Concepts and Abstract Syntax. 2014. Accessed: 2026-07-06.
https://www.w3.org/TR/rdf11-concepts/

[DVC+14] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *7th Workshop on Linked Data on the Web*. 2014. Accessed: 2026-07-06.
https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf

[FLH+16] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, G. D. Bader. Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics* 32(2):309–311, 2016.
doi:10.1093/bioinformatics/btv557

[GOM+17] R. S. Gonçalves, M. J. O'Connor, M. Martínez-Romero et al. The CEDAR Workbench: An Ontology-Assisted Environment for Authoring Metadata that Describe

Scientific Experiments. In *The Semantic Web – ISWC 2017*. Lecture Notes in Computer Science 10588, pp. 103–110. 2017.
doi:10.1007/978-3-319-68204-4_10

[HBC+21] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann. Knowledge Graphs. *ACM Computing Surveys* 54(4):1–37, 2021.
doi:10.1145/3447772

[HOD+16] J. Hastings, G. Owen, A. Dekker, M. Ennis, N. Kale, V. Muthukrishnan, S. Turner, N. Swainston, P. Mendes, C. Steinbeck. ChEBI in 2016: Improved services and an expanding collection of metabolites. *Nucleic Acids Research* 44(D1):D1214–D1219, 2016.
doi:10.1093/nar/gkv1031

[HTN+14] M. Horridge, T. Tudorache, C. Nyulas, J. Vendetti, N. F. Noy, M. A. Musen. WebProtégé: A Collaborative Web-Based Platform for Editing Biomedical Ontologies. *Bioinformatics* 30(16):2384–2385, 2014.
doi:10.1093/bioinformatics/btu256

[Jaf26] M. Jafarkhani. Connecting MetaConfigurator to the Semantic Web: RDF Authoring, Ontology Integration, and Knowledge Graph Support. Master's thesis, University of Stuttgart, 2026.
doi:10.18419/opus-18619

[JHHH15] K. Janowicz, F. van Harmelen, J. A. Hendler, P. Hitzler. Why the Data Train Needs Semantic Rails. *AI Magazine* 36(1):5–14, 2015.
doi:10.1609/aimag.v36i1.2560

[JMO+21] R. Jackson, N. Matentzoglu, J. A. Overton, R. Vita, J. P. Balhoff, P. L. Buttigieg, S. Carbon, M. Courtot, A. D. Diehl, D. M. Dooley, W. D. Duncan, N. L. Harris, M. A. Haendel, S. E. Lewis, D. A. Natale, D. Osumi-Sutherland, A. Ruttenberg, L. M. Schriml, B. Smith, C. J. Stoeckert, N. A. Vasilevsky, R. L. Walls, J. Zheng, C. J. Mungall, B. Peters. OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database* 2021:baab069, 2021.
doi:10.1093/database/baab069

[KJP+20] N. Kochev, N. Jeliazkova, V. Paskaleva, G. Tancheva, L. Iliev, P. Ritchie, V. Jeliazkov. Your Spreadsheets Can Be FAIR: A Tool and FAIRification Workflow for the eNanoMapper Database. *Nanomaterials* 10, 2020.
doi:10.3390/nano10101908

[Kou19] V. Koutkias. From data silos to standardized, linked, and FAIR data for pharmacovigilance: current advances and challenges with observational healthcare data. *Drug safety* 42(5):583–586, 2019.
doi:10.1007/s40264-018-00793-z

[lin26] linkeddata. rdflib.js Documentation. 2026. Accessed: 2026-07-06.
https://linkeddata.github.io/rdflib.js/doc/

[MM04] F. Manola, E. Miller. RDF Primer. 2004. Accessed: 2026-07-06.
https://www.w3.org/TR/rdf-primer/

[MSH+26] S. Moxon, H. Solbrig, N. L. Harris et al. LinkML: An Open Data Modeling Framework. *GigaScience* 15:giaf152, 2026.
doi:10.1093/gigascience/giaf152

[NBX+24] F. Neubauer, P. Bredl, M. Xu, K. Patel, J. Pleiss, B. Uekermann. MetaConfigurator: A User-Friendly Tool for Editing Structured Data Files. *Datenbank-Spektrum* 24:161–169, 2024.
doi:10.1007/s13222-024-00472-7

[NEB+26] F. Neubauer, K. Endo, F. Bender, E. Ciftci, N. Hansen, S. Krause, B. Uekermann, J. Pleiss. Data Management and Analysis of Metal–Organic Framework Synthesis Using Data Models. *Journal of Chemical Information and Modeling*, 2026.
doi:10.1021/acs.jcim.6c00542

[Neu26] F. Neubauer. Replication Data for: AI-Assisted RDF Authoring from JSON Data. 2026.
doi:doi:10.18419/DARUS-6299

[NPU25] F. Neubauer, J. Pleiss, B. Uekermann. Data Model Creation with MetaConfigurator. In *Datenbanksysteme für Business, Technologie und Web (BTW 2025)*. Lecture Notes in Informatics (LNI), Proceedings P-361. Gesellschaft für Informatik, Bonn, 2025.
doi:10.18420/BTW2025-60

[NUP25] F. Neubauer, B. Uekermann, J. Pleiss. AI-assisted JSON Schema Creation and Mapping. In *28th ACM/IEEE Int. Conf. Model Driven Eng. Languages and Systems Companion (MODELS-C)*. Pp. 79–83. 2025.
doi:10.1109/MODELS-C68889.2025.00019

[OBO25] OBO Foundry. COB: Core Ontology for Biology and Biomedicine. OBO Foundry ontology, 2025. Accessed: 2026-07-06.
https://github.com/OBOFoundry/COB

[Pat19] J. Patel. Bridging data silos using big data integration. *International Journal of Database Management Systems* 11(3):01–06, 2019.
doi:10.5121/ijdms.2019.11301

[Pro] O. Project. OpenRefine. Accessed: 2026-07-06.
https://openrefine.org/

[PS13] E. Prud'hommeaux, A. Seaborne. SPARQL 1.1 Overview. 2013. Accessed: 2026-07-06.
https://www.w3.org/TR/sparql11-overview/

[QUD25] QUDT.org. QUDT: Quantities, Units, Dimensions and Data Types Ontologies. Version 3.1.4, 2025. Accessed: 2026-07-06.
https://qudt.org/

[RDF] RDFLib Project. RDFLib: A Python library for working with RDF. Accessed: 2026-07-06.
https://rdflib.dev/

[Red] Redland RDF Project. Redland RDF Libraries. Accessed: 2026-07-06.
http://librdf.org/

[SBL+26] P. Stroemert, H. Borgelt, D. Linke, M. Doerr, B. Katabathuni, O. Koepler, N. Kockmann. ChemDCAT-AP: Enabling Semantic Interoperability with a Contextual Extension of DCAT-AP. arXiv:2602.01822, 2026. NFDI4Chem and NFDI4Cat. Accessed: 2026-07-06.
https://arxiv.org/abs/2602.01822

[SC26] A. Sommer, N. Car. pySHACL: A Python validator for SHACL. 2026. Version 0.31.0. Accessed: 2026-07-06.
doi:10.5281/zenodo.4750840

[SCK+05] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. L. Rector, C. Rosse. Relations in biomedical ontologies. *Genome Biology* 6(5):R46, 2005.
doi:10.1186/gb-2005-6-5-r46

[SKB+20] C. Steinbeck, O. Koepler, F. Bach, S. Herres-Pawlis, N. Jung, J. C. Liermann, S. Neumann, M. Razum, C. Baldauf, F. Biedermann et al. NFDI4Chem – Towards a National Research Data Infrastructure for Chemistry in Germany. *Research Ideas and Outcomes* 6:e55852, 2020.
doi:10.3897/rio.6.e55852

[SKL14] M. Sporny, G. Kellogg, M. Lanthaler. JSON-LD 1.0: A JSON-based Serialization for Linked Data. 2014. Accessed: 2026-07-06.
https://www.w3.org/TR/json-ld/

[USC20] USC ISI. Karma: A Data Integration Tool for Mapping Structured Data to RDF. 2020. Accessed: 2026-07-06.
https://usc-isi-i2.github.io/karma/

[Vc26] R. Verborgh, contributors. SPARQL.js. 2026. Accessed: 2026-07-06.
https://www.npmjs.com/package/sparqljs

[VKM+25] J. Villamar, M. Kelbling, H. L. More, M. Denker, T. Tetzlaff, J. Senk et al. Metadata practices for simulation workflows. *Scientific Data* 12:942, 2025.
doi:10.1038/s41597-025-05126-1

[W3C17] W3C. Shapes Constraint Language (SHACL). 2017. Accessed: 2026-07-06. https://www.w3.org/TR/shacl/

[WAB+25] S. R. Wilkinson, M. Aloqalaa, K. Belhajjame, M. R. Crusoe, B. de Paula Kinoshita, L. Gadelha, D. Garijo, O. J. R. Gustafsson, N. Juty, S. Kanwal, F. Z. Khan, J. Köster, K. Peters-von Gehlen, L. Pouchard, R. K. Rannow, S. Soiland-Reyes, N. Soranzo, S. Sufi, Z. Sun, B. Vilne, M. A. Wouters, D. Yuen, C. Goble. Applying the FAIR Principles to computational workflows. *Scientific Data* 12(1):328, 2025. doi:10.1038/s41597-025-04451-9

[WAR+26] K. Wang, D. D. Ahn, N. Rampal, J. Thomassian, N. S. Sabeva, O. Kannan, C. Borgs, J. T. Chayes, O. M. Yaghi. Synthesis of Highly Crystalline Covalent Organic Frameworks Using Large Language Models. *Journal of the American Chemical Society* 148(9):10094–10104, 2026. doi:10.1021/jacs.5c23233

[WBB+21] C. Wulf, M. Beller, T. Boenisch, O. Deutschmann, S. Hanf, N. Kockmann, R. Kraehnert, M. Oezaslan, S. Palkovits, S. Schimmler, S. A. Schunk, S. Wollny, A. Schönweiz. A Unified Research Data Infrastructure for Catalysis Research – Challenges and Concepts. *ChemCatChem* 13(14):3223–3236, 2021. doi:10.1002/cctc.202001974