

Towards Object-Oriented Linked Data Profiles for Material Science

Simon Stier^{1,*}, Lukas Gold¹, Matthias Albert Popp¹, Simon Feiler¹, Gerhard Goldbeck², Yoav Nahshon³, Andreas Räder¹, Jesper Friis⁴, Simon Clark⁶ and Patrick Lambrix⁵

¹Fraunhofer Institute for Silicate Research ISC, Neunerplatz 2, 97082 Würzburg, Germany

²Goldbeck Consulting Ltd, Cambridge, United Kingdom

³Fraunhofer Institute for Mechanics of Materials IWM, Freiburg, Germany

⁴Department of Materials and Nanotechnology, SINTEF, Norway

⁵Linköping University and Swedish e-Science Research Center, Linköping, Sweden

⁶SINTEF Industry, Trondheim, Norway

Abstract

Materials science produces data whose structure is easy to store but whose meaning is hard to share, because the same quantity, process, or dataset is described differently by every ontology, schema language, and lab information system. Object-Oriented Linked Data (OO-LD) addresses this by writing the data structure and its meaning into a single document that ordinary software tools can already read: the document is at once a JSON Schema, which validates the data, and a JSON-LD context, which says what the data means in terms of an ontology. This paper asks what such documents should look like for materials science, and proposes *profiles* in two senses. First, a layered application profile fixes the conventions science needs on top of the generic notation, and domain packages build materials-specific schemas from those definitions. Second, mapping profiles let overlapping ontologies coexist: one schema describes the data, carries a default mapping for readers with no particular preference, and records each community's alternative terms alongside it as structured annotations that can be exchanged as standard mapping sets. Choosing a community's mapping then exports the very same data file as, for example, EMMO- or PMDco-flavored RDF, without editing the file. We illustrate this with recurring patterns, including measured values with units and uncertainty, composite characteristics, processes, and datasets with exchangeable storage modes, and describe what is generated from the schemas: program code, user interfaces, and interfaces for software agents.

Keywords

linked data, JSON Schema, JSON-LD, application profiles, ontology mapping, SSSOM, quantities and units, datasets, materials science and engineering

1. Introduction

Semantic technologies and everyday research software still live in different worlds. On one side, the materials science and engineering (MSE) community has built expressive ontologies; on the other, labs run electronic notebooks, Python scripts, and web services whose native data language is JSON. Every bridge between the two is hand-crafted, and committing a schema to one ontology puts the tools and collaborators aligned with a different one at a disadvantage. Ontology matching and alignment address part of this problem, and we build on their results [19], but an alignment computed between two ontologies does not by itself tell a JSON-based application which key to write or let a validator check the document it produces. Alignments have to be recorded where developer tooling reads them.

Consider a battery researcher who records a cycling experiment in an electronic notebook, which exports JSON. Colleagues in the same project describe the same quantities with EMMO, a partner site

SeMatS 2026: The 3rd International Workshop on Semantic Materials Science co-located with the 25th International Semantic Web Conference (ISWC 2026), October 25th, 2026, Bari, Italy

*Corresponding author.

✉ simon.stier@isc.fraunhofer.de (S. Stier)

ORCID: 0000-0003-0410-3616 (S. Stier); 0000-0001-7444-2969 (L. Gold); 0000-0001-7620-1245 (M. A. Popp); 0000-0003-3565-9430 (S. Feiler); 0000-0002-4181-2852 (G. Goldbeck); 0000-0003-4774-8764 (Y. Nahshon); 0000-0002-2673-0832 (A. Räder); 0000-0002-1560-809X (J. Friis); 0000-0002-8758-6109 (S. Clark); 0000-0002-9084-0470 (P. Lambrix)



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

uses PMDco, and a data portal expects a third vocabulary. Today each of those readings requires its own export path, and the researcher sees none of the benefit. We write the data once and let each community read it in its own terms.

The Object-Oriented Linked Data (OO-LD) schema [1] connects the two worlds by combining two existing standards instead of inventing a new notation: an OO-LD document is simultaneously a valid JSON Schema [2] describing structure and a referenceable JSON-LD context [3] carrying semantics. This paper builds on that foundation and asks what a *profile* of OO-LD for materials science should look like. Our contributions are:

1. a layered profile architecture separating the generic core notation, a scientific profile, and MSE domain packages (Section 4);
2. the principle of *one syntax, tiered semantics*: consensus mappings in the context, community synonyms as structured annotations with SSSOM round-trip, and mapping-set selection producing community-flavored RDF from unchanged instances - enabling automated syntactic-to-semantic, syntactic-to-syntactic, and semantic-to-semantic transformations (Sections 4-5);
3. indicative schema patterns for quantity values, aggregated characteristics, processes, and datasets (Section 5), and the assets auto-generated from them: code, user interfaces, and agent-accessible APIs (Section 6).

This paper addresses the schema and content layer, i.e. how data and its meaning are described. The complementary exchange and discovery layer, i.e. how described assets become visible and accessible across institutions, is beyond the scope of this paper and treated in separate work on the federated data space infrastructure of the Materials Commons [4].

2. Related Work

Vocabularies and packaging. Our dataset pattern anchors to DCAT, the W3C Data Catalog Vocabulary, and its European application profile DCAT-AP, whose domain extensions (GeoDCAT-AP, MLDCAT-AP) also serve as the methodological precedent for profiling [5, 6, 7]. Tabular structure and column mapping reuse CSVW (CSV on the Web) [8]; packaging reuses RO-Crate and its `.eln` profile for electronic lab notebook (ELN) exports [9, 10]. Quantities and units anchor to QUDT (Quantities, Units, Dimensions and Types) [11]; the units ontology OM [12] pioneered the per-quantity lists of common units that our generated enums mirror (Section 5.1). Provenance and sensor observations anchor to the W3C ontologies PROV and SOSA/SSN [13, 14].

Schema languages. LinkML [15] defines its own notation with converters into the semantic web stack; the NOMAD schema system's hierarchical Metainfo [16] carries relation metadata connectable to ontologies such as EMMO; Dataspecer [17] derives JSON Schema and JSON-LD context as separate artifacts from one conceptual model; DLite/SOFT7 [18] describes dense array layouts. Semantic modeling of battery data in general has been demonstrated in an ontology-based production data space, though with custom schema languages and vocabularies [21]. OO-LD differs in that it does not introduce a notation: any JSON Schema or JSON-LD tool processes the same document, and DLite is reused as a storage descriptor rather than replaced (Section 5.4).

MSE ontologies. EMMO [22] and PMDco [23] are the reference ontologies our mappings target; the EMMO-based battery ontology BattINFO [24] supplies the cell and cycling terminology behind the battery examples. Prior SeMatS work motivates our profile mechanics directly: the ontology landscape survey of Norouzi et al. [25] documents the very multiplicity of overlapping MSE ontologies that mapping profiles are designed to live with; Andersson et al. [26] model units of measure across standards and history, a problem we meet at the pragmatic level of unit IRIs and generated enums; Del Nostro et al. [27] discuss desirable top-level-ontology characteristics, complementary to our schema-layer perspective; and Li et al. [28] argue the FAIR data management position that our implementation

concretizes, introduced for the materials design domain by Lambrix et al. [29]. Mappings themselves are exchanged as mapping sets in SSSOM, the Simple Standard for Sharing Ontological Mappings [30].

3. OO-LD in a Nutshell

An OO-LD document is a JSON Schema whose root additionally carries a JSON-LD `@context` mapping the schema's property names to IRIs (Internationalized Resource Identifiers):

```
{
  "@context": {
    "schema": "http://schema.org/",
    "name": "schema:name"
  },
  "title": "Person",
  "type": "object",
  "properties": {
    "name": { "type": "string" }
  }
}
```

Listing 1: A minimal OO-LD document. Read as a JSON Schema, it says an instance is an object with a string-valued name. Read as a JSON-LD context, the same document says that name stands for `schema:name`, so the two readings describe one and the same key.

Validation uses the document as plain JSON Schema; export to RDF, the Resource Description Framework, applies the same document as a remote JSON-LD context. Composition follows both standards at once: every `$ref` aggregation in the schema is mirrored in the context, so a schema hierarchy remains consumable by JSON-Schema-only and JSON-LD-only tools alike [1]. Two extensions matter here: `x-oold-range` constrains IRI-valued properties to a target schema, the analog of a range constraint in the ontology languages OWL and SHACL, and the term-mapping mechanism of Section 4 handles synonyms. The full notation and its standard extensions are defined in the OO-LD specification (<https://github.com/OO-LD/oold-schema>) [1].

4. A Scientific Profile of OO-LD

Layering (Figure 1). The generic OO-LD notation evolves in its own upstream specification and is pinned by version. The scientific profile adds what science needs without touching the core: reference patterns (Section 5), naming and composition conventions, and validation requirements; conformance is declared via a profile IRI (`dct:conformsTo`), analogous to DCAT-AP profiling DCAT and formalized by the W3C Profiles Vocabulary [31]. Domain packages specialize the profile for materials subdomains by importing the more fundamental definitions beneath them: the profile contributes, for example, the encoding of a quantity value and specific quantities and units; a domain package aggregates those into domain schemas, such as a tabular cycling dataset for electrochemistry that reuses the time, voltage, and current definitions (Section 5.2), or a battery specification carrying physical parthood and chemical composition; such packages combine further, up to a complete battery research dataset. Generic needs discovered in profile work are routed upstream as proposals rather than forked locally.

One syntax, tiered semantics. Communities will not converge towards one single ontology, and a schema that commits to a single ontology excludes the users of all others. A default reading can still be chosen. The profile calls it the *consensus mapping*: it is what a consumer gets when it selects no mapping set, and it draws on general, widely deployed vocabularies so that every term emits some RDF. The name records that those vocabularies are broadly usable, not that the communities have settled on one ontology. Community readings layer on top as mapping sets, and none of them is privileged. The profile therefore fixes a single syntactical schema per pattern and tiers its semantics:

- the default `@context` carries the *consensus* mapping, chosen from general, widely deployed vocabularies;
- every further community mapping lives in the structured `x-oold-context` annotation: per term, a map from synonym IRI to a promotable JSON-LD term definition plus an `x-oold-sssom` block, which round-trips to a standard SSSOM mapping set;
- selecting a mapping set *promotes* its synonyms into the context, yielding an *effective schema* that differs from the declared one only in its `@context`. The instance document is untouched; only its RDF reading changes.

The selection target is called a *target profile*. It is expressed either as mapping-set identifiers or as vocabulary namespaces, and in both cases the profile may be an *ordered* list: a consumer states that the EMMO set is preferred but the PMDco set is acceptable where EMMO has no term, or equivalently that EMMO namespaces outrank general ones. Selection then takes the highest-priority match per term, and a term with no synonym in the target keeps its consensus mapping, so every reading remains complete.

The same mechanism accommodates terms for which no published vocabulary offers a suitable IRI yet. The profile mints a local IRI for such a term, marks it as temporary and records it like any other mapping, so a work-in-progress schema is usable immediately. When a community term appears, the local IRI is superseded by a synonym entry and only the schema changes, never the instance data.

The mapping predicates are SKOS, not OWL. A synonym recorded with `skos:exactMatch` asserts that two terms are interchangeable for data exchange, not that they are logically equivalent; nothing here licenses a reasoner to conclude `owl:equivalentProperty`. That keeps the profile mechanism reasoner-safe, and it lets a mapping record a partial correspondence: where a target ontology needs a shape the flat schema does not produce, the entry is a `skos:closeMatch` carrying the reason, instead of an equivalence that would be wrong.

Each mapping set is written by the community that wants its own reading of the data, reviewed like any other schema artifact, and versioned with the profile. Where two communities claim the same term they do so in separate sets, so the consumer's selection decides between them and no central arbiter is needed. Alongside `predicate_id`, SSSOM defines a `mapping_justification` and an optional `confidence`, which together say how a synonym was arrived at and how far it can be trusted: a lexical match can be marked provisional where a curated one is not.

On terminology: a *mapping* binds one schema key to one ontology IRI; a *mapping set* (SSSOM's term, called an alignment in ontology matching) groups such bindings per community. Note that `mapping_set_id` is set-level metadata in SSSOM; the annotation denormalizes it per term so that a

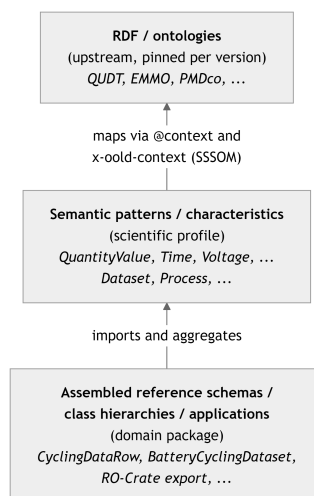


Figure 1: Layering of the profile (ontologies at the top): applications assemble reference schemas and class hierarchies from the profile's semantic patterns and characteristics, which in turn map to RDF ontologies via the consensus mapping and mapping-set synonyms.

term definition stays self-contained. Because bindings are per term, both the consensus mapping and each mapping set may draw on several vocabularies at once. Not every key needs a binding: terms without a synonym in the selected set fall back to the consensus context, and keys without any binding either remain purely syntactical and emit no RDF, or fall back to a default simple keyword-based vocabulary (a JSON-LD `@vocab` base that turns plain keys into IRIs).

Fixing the syntactical schema does not force divergent ontology models into one shape. Where models genuinely diverge, separate OO-LD schemas remain an option. In the patterns we have mapped so far, the modeled information, the sum of objects, properties, and values, largely coincides across the target ontologies even where the graph shapes differ, which is what makes term-level mapping viable; the quantity value of Section 5.1 is the case we validated per mapping. JSON-LD's `@nest` keyword adds further freedom in distributing properties over more or fewer nodes, e.g. keeping an uncertainty record a JSON subobject while its properties land on the main quantity node where an ontology demands it.

The schema fixes the JSON keys, their nesting and their datatypes, while the context fixes what each key means. That pairing supports three kinds of transformation. In the base case, *syntactic to semantic*, a JSON instance expands to RDF through the context and returns to the same JSON shape by framing. *Semantic to semantic* starts and ends in RDF and uses the shared syntactical document as its bridge, so the same instance re-exported under another mapping set connects two ontologies that name the same things differently, with the SSSOM sets recording the correspondence. *Syntactic to syntactic* is the mirror case, with syntactic start and end points and the overlapping semantic mappings as the bridge: re-serializing is carried by the context even though both ends are ordinary JSON, which is why no case-by-case conversion code is needed.

5. Patterns

We picked two representative patterns to illustrate the approach: quantity values (Section 5.1) in depth, and datasets (Section 5.4) as the integrating pattern. Two further recurring building blocks, aggregated characteristics and processes, are sketched with minimal examples; none of this is exhaustive. The two differ in maturity. The quantity value is the pattern whose mappings we have validated per mapping set and published as reference schemas; the dataset pattern and the building blocks around it are presented here as a design, with their reference schemas still in preparation.

5.1. Quantity Values

A quantity value, a number with a unit and optionally an uncertainty, is the fundamental building block reused by every typed field. The profile models it flat, with the unit as an IRI reference:

```
{
  "$schema": "https://oo-ld.org/1.0.0-rc.3/meta/oold-meta-schema.json", // OO-LD dialect

  // 1. minimal (syntactical) information model
  "title": "QuantityValue",
  "type": "object",
  "properties": {
    "value": { "type": "number" },
    "unit": { "type": "string", "format": "iri-reference" },
    "standard_uncertainty": { "type": "number" }
  },
  "required": ["value"],

  // 2. consensus mapping (QUDT); numbers stay uncoerced: JSON numbers project to
  //    xsd:double
  "@context": {
    "QuantityValue": "qudt:QuantityValue",
    "value": "qudt:value",
    "unit": { "@id": "qudt:hasUnit", "@type": "@vocab" },
  },
}
```

```

    "standard_uncertainty": "qudt:standardUncertainty"
  },

  // 3. alternative mappings as per-term synonyms, selectable by mapping set
  "x-oold-context": {
    "value": {
      // 3a. PMDco flavoured alternative mapping
      "obo:OBI_0001937": { "x-oold-ssom": { "predicate_id": "skos:exactMatch",
        "mapping_justification": "semapv:ManualMappingCuration", "confidence": 1.0,
        "mapping_set_id": ".../mappings/pmdco.sssom.tsv" } } },
      "unit": {
        // 3b. EMMO flavoured alternative mapping
        "emmo:EMMO_eeb06032_...": { "@type": "@vocab", "x-oold-ssom": {
          "predicate_id": "skos:exactMatch",
          "mapping_set_id": ".../mappings/emmo.sssom.tsv" } } },
        "obo:IAO_000039": { "@type": "@vocab", "x-oold-ssom": {
          "predicate_id": "skos:closeMatch", // unit *label*, not the unit itself
          "mapping_set_id": ".../mappings/pmdco.sssom.tsv" } } },
        "standard_uncertainty": {
          "emmo:EMMO_bba7f0f0_...": { "x-oold-ssom": {
            "predicate_id": "skos:exactMatch",
            "mapping_set_id": ".../mappings/emmo.sssom.tsv" } } }
        // ... further synonyms for schema.org, class aliases on the type term
      }
    }
  }
}

```

Listing 2: QuantityValue base schema (abridged). QUDT serves as the consensus mapping for this pattern (quantities and units are its core domain); PMDco/OBI and EMMO synonyms are stored in x-oold-context with SSSOM metadata.

We validated each design decision against the ontology sources. The unit is an IRI pointing to a unit individual, which sidesteps the question whether a unit is referenced as a class or as an individual (known in OWL as punning; unproblematic for data exchange, though description-logic reasoners treat the class and the individual as separate entities). The property is coerced with `@type: @vocab`, so that per-quantity subclasses can declare their unit individuals as context terms and pin them with an enum, which also prevents invalid IRIs caused by typos. Uncertainty is kept to the two flat fields that released vocabularies map, standard and relative standard uncertainty, following the conventions of the GUM, the metrological Guide to the Expression of Uncertainty in Measurement [32]; the full metrological record is deferred to a future node-shaped characteristic. The pattern also fixes the minimal information for a quantity: only `value` is required, `unit` and the uncertainty fields are optional, and extension happens by subclassing via `allOf`. A subclass then only restricts and defaults:

```

{
  "$id": "Time.schema.json",
  "allOf": [{ "$ref": "QuantityValue.schema.json" }],
  "@context": ["QuantityValue.schema.json",
    { "Time": "qkind:Time", // readable value terms
      "second": "qunit:SEC", "milli_second": "qunit:MilliSEC", "hour": "qunit:HR" }],
  "properties": {
    "type": { "default": ["Time"] },
    "unit": { "enum": ["second", "milli_second", "hour"], "default": "second" }
  },
  "x-oold-context": { // synonyms of the value terms
    "second": { "emmo:Second": { ... }, "obo:UO_0000010": { ... } }
  }
}

```

Listing 3: Per-quantity subclass with unit individuals as value terms; the enum is generatable from QUDT's applicable-unit inventory. Because the terms are values rather than properties, the same synonym mechanism aliases them, which is how `second` reads as `emmo:Second` under the EMMO set.

The result is that one instance,

```
{ "type": ["Time"], "value": 12.7, "unit": "second", "standard_uncertainty": 0.05 }
```

exports as three different community graphs depending only on the selected mapping set. Since `type` and `unit` carry schema defaults, an OO-LD-aware loader lets instances omit them entirely, shrinking records further toward `{ "value": 12.7 }`.

```
# consensus (QUDT)
<m1> a qkind:Time ;                                # kind individual punned as class
      qudt:value 12.7 ;
      qudt:hasUnit qunit:SEC ;
      qudt:standardUncertainty 0.05 .

# mapping set: emmo (1.0.4 development branch)
<m1> a emmo:Time ;
      emmo:hasReferencePart emmo:Second ; # generic part relation, no flat unit term
      emmo:standardUncertainty 0.05 ;
      qudt:value 12.7 .                    # no EMMO value term, consensus fallback

# mapping set: pmdco
<m1> a qkind:Time ;                                # no PMDco quantity class, fallback
      obo:OBI_0001937 12.7 ;                  # has specified numeric value
      obo:IAO_0000039 obo:UO_0000010 ;        # unit label: UO second
      qudt:standardUncertainty 0.05 .          # no flat PMDco target, consensus fallback
```

Listing 4: The same instance document under three mapping profiles. Terms without a synonym in the selected set fall back to the consensus mapping. Readable EMMO labels abbreviate the hashed IRIs (`emmo:Time` = `emmo:EMMO_d4f7d378_...`); numeric literals are shortened.

The example also shows the limits of pure term mapping: nested ontology shapes, such as the value node of released EMMO 1.0.3, cannot be produced by JSON-LD term definitions; such mappings are delegated to the SSSOM/rule level until ontology evolution removes the friction.

5.2. Aggregated Characteristics

Composite characteristics group quantity values into reusable records, here the typed row of the battery cycling example that Section 5.4 builds on:

```
{
  "title": "ElectrochemicalCyclingDataRow",
  "type": "object",
  "properties": {
    "test_time": { "$ref": "Time.schema.json" },           // Sec. 5.1 subclasses
    "voltage":   { "$ref": "Voltage.schema.json" },
    "current":   { "$ref": "ElectricCurrent.schema.json" }
  }
}
```

Listing 5: A composite characteristic. Each field reuses a quantity subclass of Section 5.1 by `$ref`, so the row inherits their units and mappings.

Row fields may themselves be composites or lists of composites (e.g. a list of air-quality readings, each holding pressure and temperature); such rows flatten deterministically into multi-index tables, e.g. columns (`"", voltage`), (`"readings.0", pressure`), ..., where units are per-column defaults in a header level rather than repeated per cell. The same mechanism covers n-dimensional data semantically: an image is a list of typed pixels, a tomogram a list of voxels; the dense array is only a storage layout (Section 5.4).

5.3. Processes

Processes anchor to PROV: a generated dataset links to its laboratory process via `generated_by` (`prov:wasGeneratedBy`), whose inverse output lists the generated data; the sample under study attaches as one of the process inputs (`prov:used`), and protocols and instruments in the same way. A minimal instance (its schema follows the same pattern as the row schema in Section 5.2):

```
{
  "@id": "test-17",
  "type": ["ElectrochemicalTest"],           // a LaboratoryProcess (prov:Activity)
  "inputs": ["cell42"],                     // the measured cell (prov:used)
  "output": "dataset-cell42"                // generated data: a Dataset instance (Sec. 5.4)
}
```

Listing 6: A minimal laboratory process, linking the sample it consumed to the dataset it produced.

An experiment is distributable in two ways: as a *payload dataset*, whose data is the process description itself, or as a *package dataset* about the process, bundling its generated outputs; the package form maps directly to an RO-Crate with a main entity and its parts.

5.4. Datasets

A dataset couples metadata with an interpreted payload (Figure 2), and the type of the payload is the schema; there is no separate schema object. Three storage modes serve the same content: (A) inline, the payload embedded in the document; (B) linked, the payload behind a JSON-LD `@id` reference that the loader dereferences; (C) file-backed, the payload in an external serialization described by a distribution that carries location, media type, an optional selector into containers (sheet, HDF5 path, SQL table), and, for dense layouts, a reused DLite data model as storage descriptor. Remote, non-file resources are data services. Minimal example of the three modes for the same battery cycling dataset:

```
// A inline: rows embedded
{ "$schema": ".../ElectrochemicalCyclingDataset", "data": [ { "test_time": { "value": 0.1 },
  "voltage": { "value": 3.62 }, "current": { "value": 0.05 } } /* ... */ ] }

// B linked: rows behind a JSON-LD reference, dereferenced by the loader
{ "$schema": ".../ElectrochemicalCyclingDataset", "data": { "@id": "cell42-rows.data.json" }
↪ }

// C file-backed: rows materialized from a described distribution
{ "$schema": ".../ElectrochemicalCyclingDataset", "distributions": [ {
  "download_url": "cell42-raw.csv", "media_type": "text/csv",
  "table_schema": ".../ElectrochemicalCyclingDataRow" } ] } // csvw:tableSchema = row
↪ schema;

// dct:conformsTo carries the dataset schema
```

Listing 7: One dataset in three storage modes. The rows materialize identically in each; only where the payload lives differs.

All three materialize identical typed rows (test time, voltage, current as quantity values from Section 5.1) and package losslessly as an RO-Crate. Mode C is what keeps the pattern usable at instrument scale: a cycling dataset of a million rows travels as a file, either the one the instrument wrote or, preferably, an already normalized serialization, described by a distribution and a row schema, so the graph stays proportional to the metadata and not to the measurements. Mode A inlines the payload and is meant for small records. The same reasoning covers position-encoded data such as dense arrays, where a storage descriptor, here a reused DLite data model, records the layout instead of the profile emitting one triple per element.

A minimal RO-Crate 1.3 package with the producing process as its main entity (the package form of Section 5.3); no new edges are introduced: the crate's `hasPart` tree is exactly the projection of

the model's parts and distributions edges, with files as leaves, and every node reuses an entity defined above:

```
{
  "@context": [ "https://w3id.org/ro/crate/1.3/context",
    { "ElectrochemicalTest": ".../ElectrochemicalTest",          // domain terms resolve
      "ElectrochemicalCyclingDataset": ".../ElectrochemicalCyclingDataset",
      "BatteryCell":                                             // real BattINFO-stack IRI
        "https://w3id.org/emmo/domain/battery#battery_68ed592a_..." } ],
  "@graph": [
    { "@id": "ro-crate-metadata.json", "@type": "CreativeWork",
      "conformsTo": { "@id": "https://w3id.org/ro/crate/1.3" },
      "about": { "@id": "." } },
    { "@id": ".", "@type": "Dataset",
      "mainEntity": { "@id": "#test-17" },                      // the package dataset (Sec. 5.3)
      "hasPart": [ { "@id": "#dataset-cell42" } ] },            // about: the process
                                                                // parts -> hasPart
    { "@id": "#test-17",
      "@type": ["CreateAction", "ElectrochemicalTest"],          // custom type persists (Sec. 5.3)
    } ]
}
```

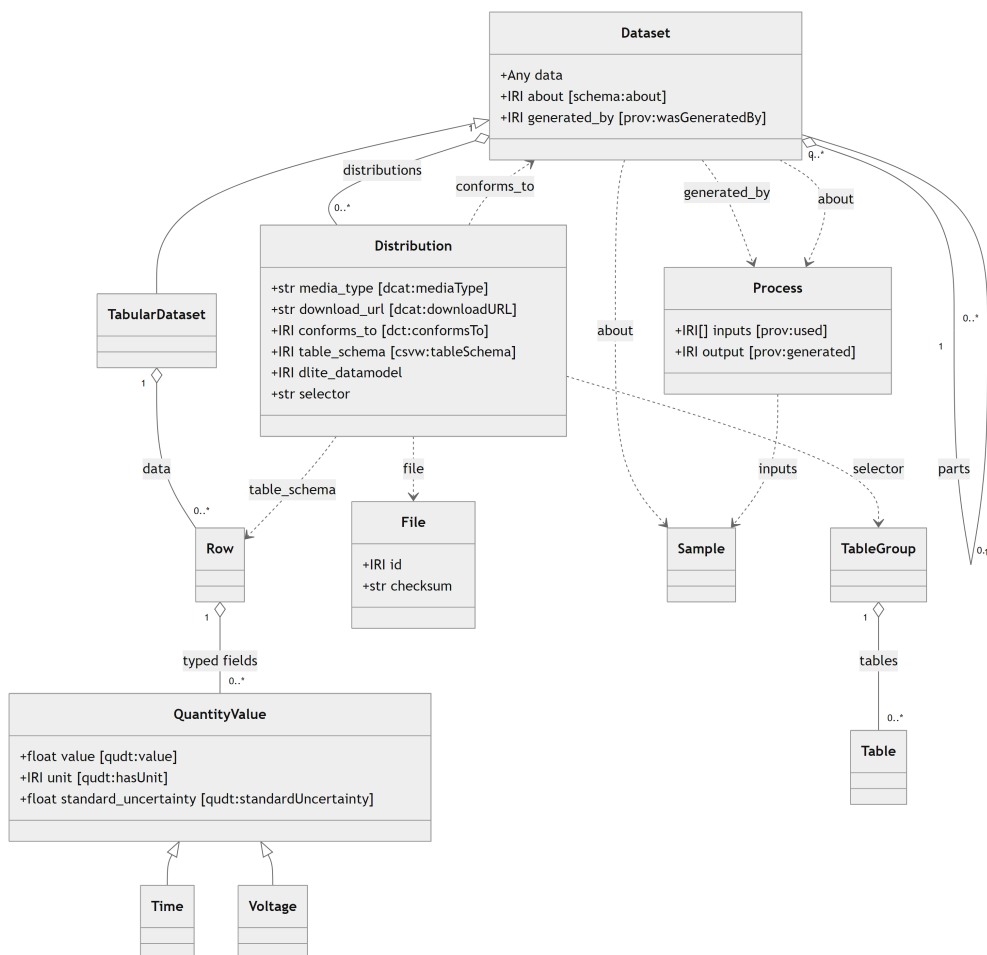


Figure 2: Core classes of the patterns and how they connect: typed rows of quantity values form tabular datasets (Sections 5.1-5.2), processes generate datasets (Section 5.3), and distributions serialize them (Section 5.4). Minimal provenance links the process to the sample it acts on (inputs) and datasets to their subject (about): the package dataset is about the process, the measurement dataset about the measured cell. Bracket annotations name each attribute's consensus mapping (Section 4); conforms_to carries the dataset schema, table_schema the row schema. A distribution references a file rather than being one, since the same file may represent more than one dataset; containers are reached through the selector, and parts nests sub-datasets.

```

    "object": { "@id": "#cell42" }, // inputs: the measured cell
↪ (prov:used)
    "result": { "@id": "#dataset-cell42" } }, // output, inverse of generated_by
    { "@id": "#cell42",
      "@type": ["Thing", "BatteryCell" ] }, // the sample, typed by BattINFO
    { "@id": "#dataset-cell42",
      "@type": ["Dataset", "ElectrochemicalCyclingDataset"], // the cycling dataset (Sec.
↪ 5.4)
    "about": { "@id": "#cell42" }, // the measurements describe the cell
    "conformsTo": { "@id": ".../ElectrochemicalCyclingDataset" } }, // the dataset schema
↪ ($schema)
    "hasPart": [ { "@id": "cell42-raw.csv" } ] }, // distributions -> File leaf
    { "@id": "cell42-raw.csv", "@type": "File", // the mode C distribution
      "conformsTo": { "@id": ".../ElectrochemicalCyclingDataRow" } } // row schema (Sec.
↪ 5.2); crate-native
// stand-in for csvw:tableSchema
  ]
}

```

Listing 8: The same content packaged as an RO-Crate. Every node reuses an entity from Listings 6 and 7; hasPart is the projection of the model's parts and distributions edges.

Custom types persist through the packaging round-trip: the @type arrays combine the RO-Crate vocabulary type with the domain class, made resolvable by the context extension; the .eln convention fixes @type to Dataset or File for filesystem nodes, so ELN exporters carry domain types via schema.org additionalType instead. Identifier strings like test-17 or dataset-cell42 are deliberately treated as opaque: the types and relationships in the graph, not naming conventions, carry the semantics.

Catalog-level concerns, i.e. how such datasets and services are discovered and exchanged across institutions, are the subject of complementary work [4].

Figure 2 adds no new concepts; it consolidates the entities introduced by the listings of Sections 5.1 to 5.4 into a single class view.

6. Generated Assets

Figure 3 places the OO-LD document at the center of four surfaces it feeds. Clockwise from the top left: user-centered design, where the schema drives generated forms and visualizations and a human works with the JSON; semantic interoperability, where the same document validates as JSON Schema and expands through JSON-LD to RDF, with transformation paths to OWL and SHACL for reasoning and validation; data spaces, where the resulting graph is queried with SPARQL and published through DCAT and DSP, which is the subject of the companion paper [4]; and workflows, where scripts, simulations and physical automation read and write the same instances, with LLM agents consuming the schema as grounding. The list below names what the profile generates for these surfaces (indicative, not exhaustive):

- **Program code:** Python data classes with object-graph mapping via oold-python, including per-quantity unit enums generated from QUDT inventories; schemas regenerate from the classes, closing the round-trip.
- **User interfaces:** schema-driven forms with IRI autocompletion against a triple store, using the standard JSON Schema UI ecosystem.
- **APIs and agents:** OpenAPI signatures derive from the same schemas and are wrapped as Model Context Protocol (MCP) tools, so LLM agents consume and produce instances validated against the profile.
- **Mappings and validation:** SSSOM mapping sets export from the schema annotations; continuous validation checks schemas, instances, and mappings against the pinned ontology versions.

integration; the individual quantities and units are worked examples rather than a covering inventory, and the patterns of Sections 5.2 to 5.4 are designs whose reference schemas are still to come.

Further work is needed to complete those schemas and to harmonize the patterns with one another, in particular the complex ones such as composition and process provenance, where several patterns meet in a single document. Beyond that, whether the mechanics hold outside the vocabularies we have covered is untested, and applying them to a further ontology is the most useful next step; how far curation scales is equally open, since mapping sets are cheap to write and expensive to keep correct as their target ontologies move. Term mapping also still cannot produce a graph shape that the document does not contain, as Section 5.1 shows, and closing that gap needs rule-level mapping or a change upstream in the ontology.

Acknowledgements

The authors thank all partners of the MaterialsCommons consortium for the discussions that shaped this work. This work is part of the MaterialsCommons project. Funded by the European Union (Horizon Europe grant agreement No 101293545). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) in order to: structure content, improve writing style and citation management. All concepts, designs, and technical results presented originate from the authors. When using these tools, the authors always validated the results and take full responsibility for the publication's content.

References

- [1] Stier, S., Gold, L.: OO-LD: Object-Oriented Linked Data Schema, v1.0.0-rc.3 (2026). <https://github.com/OO-LD/oold-schema>, <https://doi.org/10.5281/zenodo.22068117>
- [2] Wright, A., Andrews, H., Hutton, B., Dennis, G. (eds.): JSON Schema: A Media Type for Describing JSON Documents. Internet-Draft draft-bhutton-json-schema-01, IETF (2022). <https://json-schema.org/draft/2020-12/json-schema-core>
- [3] JSON-LD 1.1. W3C Recommendation (2020). <https://www.w3.org/TR/json-ld11/>
- [4] Stier, S., et al.: The Federated Data Space Infrastructure of Materials Commons: Concept and Architecture. In: Proceedings of the 3rd International Workshop on Semantic Materials Science (SeMatS 2026), CEUR Workshop Proceedings (2026)
- [5] SEMIC: DCAT-AP 3.0.1 (2025). <https://semiceu.github.io/DCAT-AP/releases/3.0.1/>
- [6] SEMIC: GeoDCAT-AP 3.1.0 (2026). <https://semiceu.github.io/GeoDCAT-AP/releases/3.1.0/>
- [7] SEMIC: MLDCAT-AP 3.1.0 (2026). <https://semiceu.github.io/MLDCAT-AP/releases/3.1.0/>
- [8] CSV on the Web (CSVW): Tabular Data Model. W3C Recommendation (2015). <https://www.w3.org/TR/tabular-data-model/>
- [9] Soiland-Reyes, S., et al.: Packaging research artefacts with RO-Crate. Data Science 5(2), 97-138 (2022). <https://doi.org/10.3233/DS-210053>
- [10] The ELN Consortium: TheELNFileFormat. <https://github.com/TheELNConsortium/TheELNFileFormat> (accessed 2026-08-27)
- [11] QUDT: Quantities, Units, Dimensions and Types, v3.5.0. <https://qudt.org/>; <https://doi.org/10.25504/FAIRsharing.d3pqw7>
- [12] Rijgersberg, H., van Assem, M., Top, J.: Ontology of units of measure and related concepts. Semantic Web 4(1), 3-13 (2013). <https://doi.org/10.3233/SW-2012-0069>

- [13] Lebo, T., Sahoo, S., McGuinness, D. (eds.): PROV-O: The PROV Ontology. W3C Recommendation (2013). <https://www.w3.org/TR/prov-o/>
- [14] Haller, A., Janowicz, K., Cox, S., et al. (eds.): Semantic Sensor Network Ontology (SOSA/SSN). W3C Recommendation (2017). <https://www.w3.org/TR/vocab-ssn/>
- [15] Moxon, S., Solbrig, H., Unni, D., et al.: The Linked Data Modeling Language (LinkML): A General-Purpose Data Modeling Framework Grounded in Machine-Readable Semantics. ICBO 2021, CEUR Vol-3073, 148-151. <https://ceur-ws.org/Vol-3073/paper24.pdf>; <https://linkml.io/>
- [16] Ghiringhelli, L.M., et al.: Shared metadata for data-centric materials science. *Sci. Data* 10, 626 (2023). <https://doi.org/10.1038/s41597-023-02501-8>
- [17] Klímek, J., Stenchlák, Š., Polický, A., Škoda, P., Nečaský, M.: Towards Authoring of Vocabularies and Application Profiles using Dataspecer. ISWC 2024 Posters, Demos and Industry Tracks, CEUR Vol-3828 (2024). <https://ceur-ws.org/Vol-3828/paper33.pdf>
- [18] DLite / SOFT7 data models. <https://github.com/SINTEF/dlite>
- [19] Euzenat, J., Shvaiko, P.: *Ontology Matching*, 2nd edn. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-38721-0>
- [20] OO-LD reference schemas: modular, ontology-mapped JSON Schema and JSON-LD schemas for science and materials science and engineering. <https://schemas.oo-ld.org/>; <https://github.com/oo-ld/oold-reference-schemas>
- [21] Stier, S.P., Xu, X., Gold, L., Möckel, M.: Ontology-based battery production dataspace and its interweaving with artificial intelligence-empowered data analytics. *Energy Technology* 12(4), 2301305 (2024). <https://doi.org/10.1002/ente.202301305>
- [22] Elementary Multiperspective Material Ontology (EMMO), release 1.0.3. <https://w3id.org/emmo>; <https://github.com/emmo-repo/EMMO>. The metrology terms used in Listing 4 are not yet in a release; that mapping set pins the 1.0.4 development branch by commit.
- [23] Bayerlein, B., et al.: PMD Core Ontology: Achieving semantic interoperability in materials science. *Materials and Design* 237, 112603 (2024). <https://doi.org/10.1016/j.matdes.2023.112603>; <https://w3id.org/pmd/co>
- [24] Clark, S., Bleken, F.L., Stier, S., et al.: Toward a unified description of battery data. *Advanced Energy Materials* 12(17), 2102702 (2022). <https://doi.org/10.1002/aenm.202102702>
- [25] Norouzi, E., Waitelonis, J., Sack, H.: The landscape of ontologies in materials science and engineering: a survey and evaluation. *SeMatS 2024*, CEUR Vol-3760. <https://ceur-ws.org/Vol-3760/paper3.pdf>
- [26] Andersson, O.B., Li, H., Lambrix, P., Armiento, R.: An ontology for units of measures across history, standards, and scientific and technology domains. *SeMatS 2024*, CEUR Vol-3760. <https://ceur-ws.org/Vol-3760/paper8.pdf>
- [27] Del Nostro, P., et al.: Top level ontologies: desirable characteristics in the context of materials science. *SeMatS 2024*, CEUR Vol-3760. <https://ceur-ws.org/Vol-3760/paper9.pdf>
- [28] Li, H., Esguerra, J.L., Wang, F., Blomqvist, E.: FAIR data management for designing sustainable advanced materials: a position paper. *SeMatS 2025*, CEUR Vol-4104. <https://ceur-ws.org/Vol-4104/paper4.pdf>
- [29] Lambrix, P., Armiento, R., Delin, A., Li, H.: FAIR Big Data in the Materials Design Domain. In: *Encyclopedia of Big Data Technologies* (2022). https://doi.org/10.1007/978-3-319-63962-8_293-2
- [30] Matentzoglou, N., et al.: A Simple Standard for Sharing Ontological Mappings (SSSOM). *Database* 2022, baac035. <https://doi.org/10.1093/database/baac035>
- [31] Atkinson, R., Car, N.J. (eds.): *The Profiles Vocabulary*. W3C Working Group Note (2019). <https://www.w3.org/TR/dx-prof/>
- [32] Joint Committee for Guides in Metrology: *Evaluation of measurement data - Guide to the expression of uncertainty in measurement*. JCGM 100:2008, BIPM (2008). <https://doi.org/10.59161/JCGM100-2008E>