

Supporting the understanding of ontologies for scientific knowledge graphs with the new version of LODE

Valentina Pasqual^{1,*}, Ahmadreza Nazari² and Silvio Peroni^{1,3}

¹Digital Humanities Advanced Research Centre (/DH.arc), Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy

²Digital Humanities and Digital Knowledge, Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy

³Research Centre for Open Scholarly Metadata, Department of Classical Philology and Italian Studies, University of Bologna, Bologna, Italy

Abstract

Scientific Knowledge Graphs (SKGs) rely on increasingly complex and heterogeneous semantic models, whose reuse and interoperability require ontologies and other semantic artefacts to be understandable by both machines and humans. However, existing ontology documentation tools provide limited support for modern reuse-oriented modelling practices, where entities are distributed across imported modules and enriched through cross-module annotations. This paper presents the new version of LODE, a re-engineered and extensible framework for generating human-readable documentation of semantic artefacts. LODE preserves the established single-page, W3C Recommendation-style documentation paradigm of its legacy predecessor, and introduces a modular architecture that separates artefact interpretation (Reader), internal representation (Model), and rendering (Viewer). Served as a web service, LODE supports the documentation of OWL ontologies and enables additional documentation features (e.g., entity stand-alone documentation, RDF entity provenance, Markdown rendering). We demonstrate the applicability of LODE through the documentation of the Scientific Knowledge Graph Interoperability Framework ontology (SKG-O), showing how it improves the accessibility and reuse of modular scientific knowledge graph models.

Keywords

Ontology documentation, OWL ontologies, Web tool, Scientific Knowledge Graphs, Semantic artefacts documentation

1. Introduction

In 2009, in two seminal articles, David Shotton introduced the concept of *semantic publishing* as “anything that enhances the meaning of a published journal article, facilitates its automated discovery, enables its linking to semantically related articles, provides access to data within the article in actionable form, or facilitates integration of data between articles” [1], which can be reached by “enriching the article with appropriate metadata that are amenable to automated processing and analysis, allowing enhanced verifiability of published information and providing the capacity for automated discovery and summarization” [2]. Starting from those bases, the Semantic Web community interested in the scholarly publishing domain has largely investigated and expanded upon such a topic in a plethora of events, often organised in the context of primary research venues such as the European Semantic Web Conference (with the 2011-2014 editions of the SePublica workshop [3]), the Web Conference (with the 2015-2018 editions of the SAVE-SD workshop [4]), and the International Semantic Web Conference (with the 2021-2026 editions of the Sci-K workshop [5]).

Among the various aspects that characterise semantic publishing, ontologies, vocabularies, taxonomies, and other semantic artefacts play a prominent role in implementing a specific data model

6th International Workshop on Scientific Knowledge: Representation, Discovery, and Assessment, Oct 2026, Bari, Italy

*Corresponding author.

EMAIL: valentina.pasqual2@unibo.it (V. Pasqual); ahmadreza.nazari@studio.unibo.it (A. Nazari); silvio.peroni@unibo.it (S. Peroni)

URL: <https://www.unibo.it/sitoweb/valentina.pasqual2> (V. Pasqual); <https://www.unibo.it/sitoweb/silvio.peroni> (S. Peroni)

ORCID: 0000-0001-5931-5187 (V. Pasqual); 0009-0000-3884-1575 (A. Nazari); 0000-0003-0530-4305 (S. Peroni)



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

for the scholarly domain and, thus, in enabling semantic interoperability among systems [6, 7, 8]. Indeed, in the past, several ontologies have been proposed, such as the *Semantic Publishing And Referencing (SPAR) Ontologies* (<https://www.sparontologies.net>) [9] and the *Bibliographic Ontology (BIBO)*, (<https://www.dublincore.org/specifications/bibo/>), and such technologies have been adopted by several providers of scientific knowledge graphs (SKGs) such as OpenCitations (<https://opencitations.net>) [10] and Open Research Knowledge Graph (ORKG, <https://orkg.org/>) [11].

However, to foster broader adoption of such semantic artefacts, an important tool to provide for accompanying an ontology is *human-readable documentation* that enables an easier comprehension of the various types of entities (classes, properties, individuals, etc.) an ontology introduces in a way that is understandable by non-technical experts rather than solely knowledge engineers and semantic web developers. This is even more important when an ontology is defined by gathering together entities from existing models without redefining their original semantics, thus performing a *direct reuse* of ontological terms [12]. However, such a direct reuse can be performed by either *importing* the desired ontologies into a new one (via `owl:imports` axioms) or including only the selected ontology terms in a new ontology *referencing* the reused ontologies via `rdfs:isDefinedBy` axioms. While the former case includes the semantics of reused ontologies entirely within the new one, in the latter case the semantics of reused terms are delegated to external ontologies, avoiding the unnecessary inclusion of useless ontological entities. In these cases, the rich visualisation of annotation information for each entity in an appropriate ontology documentation is crucial as it may clarify how the various entities are used in the context of the application.

Several tools, such as LODE [13, 14] and WIDOCO [15], have been developed to create ontology documentation, often in HTML, by processing an ontology and its annotations¹. However, their development has been guided by requirements for ontology documentation that have been recently expanded, considering the importance that semantic artefacts have today, such as enabling the possibility of rendering the documentation of single entities within an ontology and the fact that the tool should be agnostic and able to represent all the annotations the entities it includes have independently of the particular annotation vocabulary adopted. Thus, such tools should be extended (or new ones should be developed) to address these needs and be continuously adopted within the scholarly domain – being them used, for instance, also to produce the documentation of the SKG-IF RDA Recommendation [16].

In this paper, to address the aforementioned needs, we introduce (for the first time) a new version of LODE (v2.0.0) [17], now maintained by OpenCitations to ensure its code is continuously developed and sustained, and to guarantee the availability of an online service accessible via a browser and a REST API, which is now included in the OpenCitations infrastructure. This new version includes a full re-engineering of the code, now available in Python, and implements several new features to address all the limitations in the old version of the system. To demonstrate its new capabilities in action, we use the SKG-IF Ontology (SKG-O, <https://w3id.org/skg-if/ontology/>) as a case study, showing improvements in its documentation using LODE 2.0, which will be adopted to provide the new documentation of SKG-O in the near future.

The rest of the paper is organised as follows. In Section 2, we introduce the most prominent related tools for the production of ontology documentation. In Section 3.1, we present the new architectural organisation of LODE 2 and its main components, while in Section 4 we present all the new features introduced in this version. Finally, in Section 5, we show some of the new features of LODE in action during the generation of the documentation for SKG-O, and in Section 6, we conclude the paper by

¹Indeed, it is worth mentioning that the first alpha (and internal) version of LODE was created to quickly obtain human-readable documentation for the first set of SPAR Ontologies under development between 2010-2011. Having documentation was crucial at that time to organise several meetings with domain experts, who lacked expertise in ontological development, to discuss various aspects of domain modelling, while masking the obscure technicalities introduced by OWL and the formats used to implement such ontologies. In particular, during these meetings, we often gathered additional information that led to slight changes to parts of the modelling, and having an automated process for creating documentation helped speed up discussions with domain experts. From that point, producing a full standalone software (<https://web.archive.org/web/20120822012904/http://sourceforge.net/projects/lode/>), also run as a Web service (<https://web.archive.org/web/20120402180140/essepuntato.it/lode/>), that may help all ontology engineers to simplify the creation of ontology documentation either for developing or publishing purposes was a natural subsequent step.

sketching out some future work.

2. Related Work

SKGs are typically designed and evolved to address specific research objectives and application scenarios. This use-case-driven development results in heterogeneous semantic models, domain-specific vocabularies, and project-dependent modelling decisions, ultimately leading to fragmentation across the SKG ecosystem [16]. A recent effort to overcome this fragmentation is the Scientific Knowledge Graph - Interoperability Framework (SKG-IF, <https://skg-if.github.io>) [16], which provides a shared interoperability layer through common concepts and mappings while preserving the autonomy of individual data models. Its conceptual model is formalised by the SKG-IF Ontology (SKG-O, <https://w3id.org/skg-if/ontology/>), thereby enabling interoperability and promoting the long-term sustainability of scientific knowledge infrastructures. Nevertheless, the practical reuse of the framework also relies on making its semantic artefacts understandable to human users.

Among currently maintained ontology documentation approaches, the *Live OWL Documentation Environment* (LODE, <https://w3id.org/lode/>) [13, 14, 18] established the reference model for human-readable ontology documentation and became a *de facto* standard for OWL ontology publication. Introduced in 2012, LODE defined the widely adopted single-page, W3C Recommendation-style representation of ontologies, automatically extracting classes, object and data properties, named individuals, annotation properties, general axioms, and namespace declarations, and presenting them through structured HTML documentation with their textual descriptions and hyperlinks. Built in Java and available on GitHub (<https://github.com/essepuntato/lode>), it linearises the ontology to RDF/XML via the OWL API (<https://github.com/owllcs/owlapi>) [19] and then applies large XSLT 2.0 stylesheets (<https://www.w3.org/TR/xslt20/>) [20] to produce HTML documents; optional parameters (e.g., `owlapi`, `imported`, `closure`, `reasoner`, `lang`) extended this pipeline. Its adoption by initiatives such as the SPAR Ontologies [9], together with its integration into publication workflows based on content negotiation for dereferenceable ontology URIs, further consolidated the use of automatically generated, single-page HTML documentation as a common practice for Semantic Web publishing. Its monolithic, XSLT-centred design, however, tightly coupled ontology parsing and graphical rendering, making the system hard to extend or modernise after a decade of technological change.

Over the years, LODE has served as the foundation for several ontology documentation tools, either by being integrated as a documentation component, as in Wizard for Documenting Ontologies (WIDOCO, <https://github.com/dgarijo/WIDOCO>) [15, 21], or by being re-engineered into new implementations, as in pyLODE (<https://github.com/rdfliib/pyLODE>) [22]. pyLODE provides a Python-based reimplementaion built on RDFLib (<https://github.com/rdfliib/rdfliib>) [23], removing the original LODE's Java/XSLT dependency and integrating ontology documentation generation into Python-oriented semantic workflows through a command-line interface, a Python library, a Docker image, and a web service (<https://tools.kurrawong.ai/pylode>). However, Abox definition (i.e., named individuals) is not included, and additional functionalities provided by LODE (e.g., imported ontology handling) are not supported.

WIDOCO, by contrast, extends the original LODE pipeline with a broader documentation workflow. It automatically enriches the generated documentation with ontology metadata and publication-oriented features, including interactive visualisation through WebVOWL (<https://github.com/visualdataweb/webvowl>) [24], provenance metadata based on PROV-O, licence information, Schema.org annotations, change logs, and content-negotiation support. Additionally, WIDOCO encourages ontology publishers to supply a rich set of metadata to maximise the completeness and quality of the resulting documentation (<https://github.com/dgarijo/Widoco/blob/master/doc/metadataGuide/guide.md>) and best practices (<https://dgarijo.github.io/Widoco/doc/bestPractices/index-en.html>), and provides rich-text annotations (Markdown rendering facilities). Like pyLODE, however, it produces a single, self-contained HTML documentation page for each ontology. WIDOCO has been used to generate the published documentation of SKG-O [16].

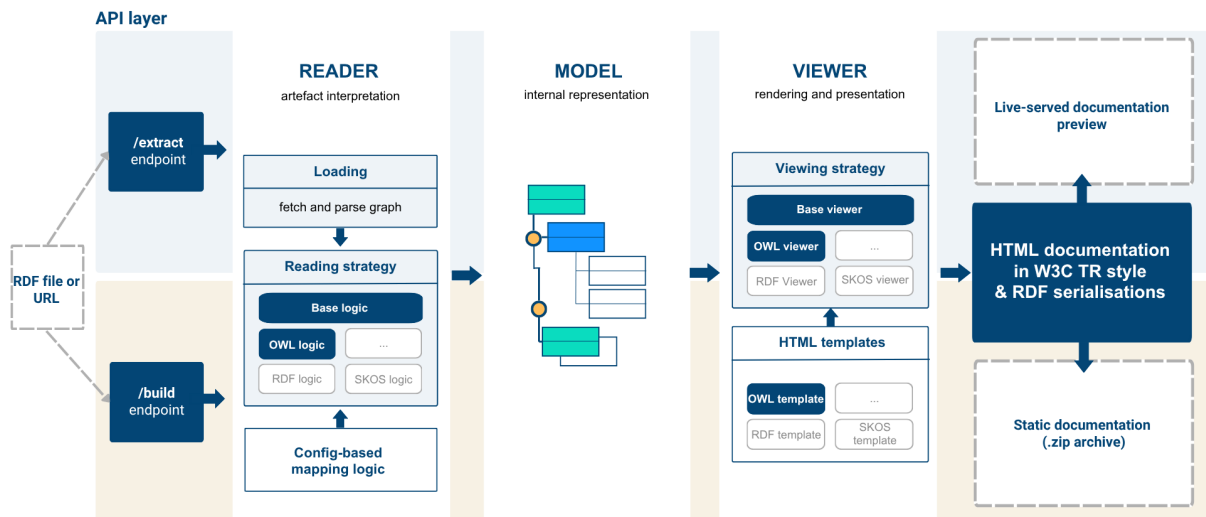


Figure 1: High-level representation of LODE pipeline

3. Architecture

The new version of LODE re-implements existing legacy features (e.g. OWL documentation on single-page and W3C-styled documentation, imported and closure handling, etc.), and extend them in a modernised architecture, replacing the XSLT pipeline with a modular Python-based environment. While the original LODE was conceived for documenting OWL ontologies, the contemporary Semantic Web landscape increasingly relies on a broader ecosystem of semantic artefacts, including RDF(S) vocabularies, SKOS concept schemes, SHACL shapes, and other knowledge representation resources. Accordingly, the new version of LODE is designed as an extensible documentation framework. OWL documentation represents the first fully supported artefact family, while the architecture is explicitly designed to accommodate additional semantic resource types through future extensions.

3.1. Internal architecture

The re-engineered version of LODE is designed around a strict separation of concerns, decoupling the system into three layers, as shown in Figure 1. The *Reader* parses the input file that defines the semantic artefact for which the documentation is to be produced, and maps it onto the internal model. The *Model* organises and collects the parsed information into Python classes, providing a uniform and independent layer through which the Reader communicates with the *Viewer*, which uses the populated model to generate the final HTML documentation.

Reader. The Reader takes as input a semantic artefact (available in RDF) via its URL or file upload (through the loader module) and loads it into an RDFLib graph. A *configuration manager* instantiates the logic layer responsible for interpreting the graph. The framework first provides a base logic that implements the processing of core RDF and RDFS constructs, which are common to semantic artefacts represented as RDF graphs. On top of this layer, family-specific logic modules introduce the additional constructs required by a given semantic artefact model.

For OWL artefacts, *OwlConfigManager* selects the OWL configuration and instantiates an *OwlLogic*, which extends the base logic with OWL-specific processing capabilities. The logic iterates over the input graph and maps triples onto the Model. This mapping is driven by a declarative

configuration file that enumerates the vocabulary terms of the family and specifies, for each term, the model class to instantiate and how to populate it. The OWL implementation covers the terms of the OWL namespace (<http://www.w3.org/2002/07/owl#>) following the constructs described in [25].

Declarative mapping is supported by a set of *handler* functions that resolve ambiguous cases from context and perform lightweight, documentation-oriented inferences, such as traversing the class hierarchy to resolve undeclared domains and ranges and assigning default values if nothing is retrieved (e.g. `owl:Thing`). When a predicate may map to more than one target class, a handler selects the target from context: for instance, `rdfs:domain` and `rdfs:range` are dispatched through dedicated handlers that also infer the kind of property involved (object, data or annotation) from how it is used if not explicitly declared in the ontology. Handlers likewise classify blank nodes according to their context, e.g. by classifying restrictions or reified annotations. Additionally, triples not covered by any configuration term are preserved as generic statements to ensure that no assertion is silently dropped (e.g. SKOS annotations). Throughout, the Reader records RDF provenance by locally storing a Model instance containing the set of triples that produced it from the source graph.

Model. The Model is a unified, extensible object-oriented schema of Python classes, as illustrated in Figure 2. Designed as a documentation-oriented representation, it is intended to support faithful documentation and *not* intended as a meta-model of the Semantic Web nor as an alternative to OWL/RDF(S) semantics. It is conceived as a shared vocabulary that can represent different types of semantic artefacts through dedicated extensions. The current implementation focuses on representing RDF/RDFS and OWL constructs, corresponding to the base layer (yellow) and the OWL-specific extension layer (green) shown in the UML diagram in Figure 2. Additional artefact-specific extensions (e.g. SKOS) are planned to be integrated within the same framework.

The structure shared by all entities (e.g. identifier, deprecation status, labels, notes, definitions, examples and provenance metadata) is represented by the top class `Resource`. Class-like resources are represented by `Concept`, which carries the subclass, equivalence and disjointness relations together with the SKOS-style mappings (broad, narrow, related, exact and close match). Properties are represented by `Property`, holding domain, range and functionality, and specialised into `Relation` for OWL object properties, `Attribute` for data properties, and `Annotation` for annotation properties. Named entities are represented by `Individual`, while ontology-level metadata (imports, version, prior versions, compatibility) are represented by `Model`; and literal values with their language and type are represented by `Literal` and `Datatype`. Complex OWL 2 class expressions are represented by a hierarchy of restriction classes rooted in `Restriction`: existential and universal restrictions by `Quantifier`, cardinality constraints by `Cardinality`, `owl:hasValue` restrictions by `Value`, boolean combinations by `TruthFunction`, enumerations by `OneOf` and datatype facets by `DatatypeRestriction`. SWRL rules are represented by `Rule`, `Atom` and `Variable`. Generic Statements represent reification and unmapped (possibly reified) statements.

Viewer. The Viewer takes the populated Model produced by the Reader and, accessing it through its defined interface (e.g. OWL), retrieves entities and restructures the information it requires for the sake of presentation. Specifically, it orders entities deterministically so that the same artefact always yields the same output, resolves language tags to select the appropriate label and comment for the target language, and sanitises the Markdown embedded in annotations before rendering it to HTML. The Viewer further implements a client-side rendering pipeline for annotation values containing Markdown. The pipeline detects and transforms Markdown constructs (including clickable URLs, nested lists, and isolated code blocks) into HTML. The generated markup is sanitised using `marked.js` (<https://github.com/markedjs/marked>) [26] and `DOMPurify` (<https://www.npmjs.com/package/dompurify>) [27], preventing malformed or potentially unsafe annotations from affecting the documentation structure or client-side execution. In addition, it categorises entities hierarchically, reconstructing the subclass and sub-property trees. The `OwlViewer` adds the OWL-specific presentation: the entity categories that structure the documentation (concepts, objects, data and annotation properties, and individuals)

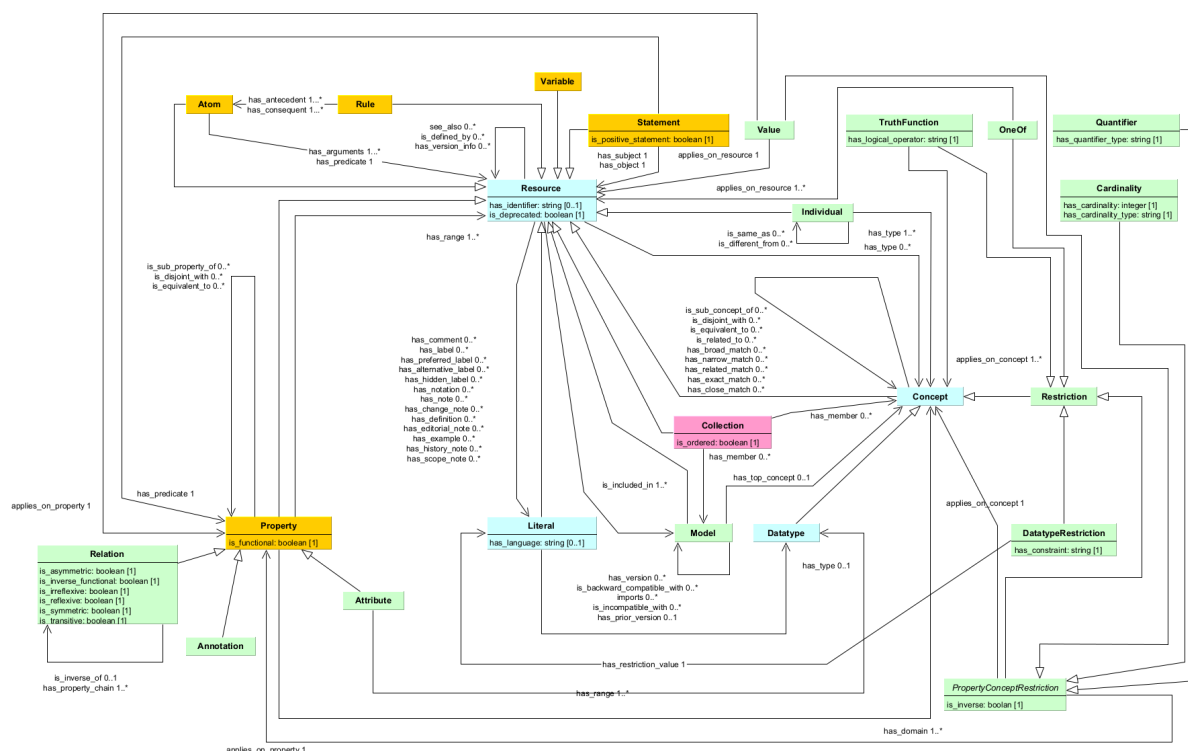


Figure 2: The UML diagram of the Python classes implementing the Model module. In particular, the classes in yellow have been derived from RDF, those in light blue from RDFS, while the classes in green and pink have been defined from OWL and SKOS, respectively.

and their order, reproducing the layout of the legacy LODE. The resulting representation is injected into a Jinja [28] template (<https://github.com/pallets/jinja>), styled with Bootstrap 5 [29] (<https://github.com/twbs/bootstrap>) for responsive layout, and uses CSS stylesheets (mimicking the legacy LODE). The Viewer supports two rendering modes: a complete single-page document listing all entities of the artefact (as done in the legacy LODE), and a per-resource view that documents an individual entity on a dedicated page.

3.2. Deployment, maintenance and hosting

LODE 2 is available at lode.opencitations.net (and accessible at <https://w3id.org/lode> via redirection). The service is exposed as a FastAPI² service through two endpoints:

- `/extract` renders a single semantic artefact on demand. It accepts the semantic artefact (via url or file upload), a `read_as` semantic artefact type (e.g. `owl` for an OWL ontology), an optional resource for single-entity mode, a `lang` filter, and the `imported/closure` switches inherited from legacy LODE. Through HTTP content negotiation, the same request returns either the rendered HTML documentation or a machine-readable serialisation (Turtle, RDF/XML, JSON-LD and N-triples), for the whole ontology (including the full set of ontology axioms) or for the selected resource (sub-graph containing all axioms explicitly asserted about that entity).
- `/build` produces a self-contained static documentation site, packaged as a `.zip` archive, for offline browsing or self-hosting. For automated and reproducible workflows (e.g. CI/CD publishing to GitHub Pages), the same static site is generated by the `lode build` command-line tool.

Critically for a scholarly-infrastructure tool, long-term maintenance and hosting are undertaken by OpenCitations – one of the infrastructures implementing SKG-IF. Persistent `w3id.org` URIs with

²<https://fastapi.tiangolo.com/>

content negotiation are being provisioned, so that ontology URIs can dereference directly to their LODE 2 documentation.

4. New Features

The new version of LODE preserves the original single-page, W3C Recommendation-style documentation paradigm, including support for imported ontologies and their closure, while extending its capabilities to support modern semantic artefacts (e.g. including user-friendly navigation for large imported ontologies and distributed annotations handling, as shown in Section 5). The main new features introduced by LODE can be summarised as follows.

Improved documentation generation and completeness. Mechanisms that enhance the quality and coverage of the generated documentation:

- *Documentation-oriented inference and default values.* A lightweight documentation-oriented inference mechanism infers missing information and applies default values to improve documentation completeness. Missing property types (object, datatype, or annotation properties) are inferred from property hierarchy relations, inverse property axioms, and RDF usage patterns (i.e. whether a property connects resources or literals). When domain or range axioms are not explicitly declared, the system recovers inherited constraints by traversing the property hierarchy and applies OWL-compatible defaults when no information is available (i.e. `owl:Thing` for object property domain and range, and `rdfs:Literal` for data property range). If no further declaration or usage pattern is available, properties are classified as annotation properties, as adopted by reference tools such as the OWL API and Protégé.³
- *Safe rich-text annotations rendering.* Markdown content is consistently rendered in all textual literals, independently of the annotation property through which they are provided. LODE adopts an annotation-agnostic approach to preserve documentation-oriented formatting, including headings, inline and block code, bullet lists, hyperlinks, and other standard Markdown constructs.

Enhanced entity-level documentation. Mechanisms that enrich the documentation provided for the documentation of individual entities:

- *Dedicated Statements block.* Building upon the entity cards introduced in its legacy version, which organises the documentation of each entity around its extracted label (displayed as the card title), IRI, and OWL axioms, each card is extended by a dedicated *Statements* section. This section exposes RDF(S)/OWL assertions that cannot be represented through predefined documentation templates. This prevents information loss by preserving entity annotations expressed through arbitrary properties, including those not explicitly supported by the platform's rendering rules, keeping them graphically separated from formal OWL axioms.
- *Per-entity RDF provenance.* Each entity card includes a collapsible *RDF Provenance* panel exposing the RDF description underlying the documented resource. By providing direct access to the associated triples in multiple serialisations (Turtle, RDF/XML, JSON-LD, and N-Triples), it enables source-level verification at entity granularity, allowing users to validate individual documentation entries without navigating the entire documentation.
- *Stand-alone entity documentation page.* Each explicitly defined entity in the ontology is associated with a dedicated link next to its IRI that links to a stand-alone documentation page for that entity. The page supports the same content-negotiation mechanisms as the full documentation and exposes the RDF sub-graph centred on the selected entity, i.e. the same description provided in

³This is a lax-parsing heuristic rather than conformant behaviour and the OWL API defaults undeclared properties to annotation properties to avoid rejecting real-world ontologies; see <https://github.com/owlcs/owlapi/issues/100> (accessed 19 September 2026).

the *RDF Provenance* panel. This enables dereferenceable, entity-level documentation, where the documented scope is the entity itself rather than the complete ontology.

Enhanced navigation and entity discovery. Mechanisms supporting the exploration of large documentation outputs:

- *Structural navigation aid.* A persistent, collapsible sidebar reproduces the complete structure of the ontology table of contents, exposing all documented sections' respective entities through a listed view to aid browsing and discovery across large documentation outputs.
- *Client-side search.* An embedded, persistent search bar enables direct retrieval of documented entities by matching entity labels, displaying the number of occurrences and dynamically synchronising with the sidebar navigation. A graphical cue highlights matching entities within the documentation body, while non-matching entries in the sidebar are visually de-emphasised.

Flexible deployment and accessibility. Mechanisms supporting reproducible generation and distribution of documentation:

- *Local artefact processing.* LODE supports the direct upload and processing of local ontology files, extending the previous URL-based workflow of legacy LODE.
- *Static documentation generation.* In addition to the live documentation endpoint inherited from its legacy version (`/extract`), LODE provides dedicated build functionality to generate self-contained static documentation sites. The generated package includes both the complete ontology documentation page and dedicated pages for each explicitly defined entity, preserving the same entity-level views available in the live version. Static sites can be produced through the web service API or the command-line interface, enabling offline browsing, self-hosting, and automated publication workflows (see Section 3.2).

5. Case Study: Documenting the SKG-IF Ontology

As claimed in Section 2, SKG-O documentation is currently generated using WIDOCO [15]. We compare the documentation generated for the same ontology using WIDOCO and the new version of LODE, highlighting differences in annotation coverage, rendering strategies, and newly introduced documentation capabilities over the chosen case study.

Figure 3 compares the current SKG-O documentation generated with WIDOCO (left) with the documentation produced by LODE (right). The WIDOCO rendering reflects the subset of ontology annotations selected through its wizard-driven workflow, as described in Section 2. Conversely, LODE adopts an annotation-agnostic rendering approach, exposing the complete set of available ontology-level metadata (e.g. dates, preferred namespace prefixes, and namespace URIs in Figure 3) without requiring these elements to be mapped to predefined documentation templates.

SKG-O adopts a modular architecture in which its six core entities are organised into dedicated ontology modules formally imported by the core ontology, namely agent (<https://w3id.org/skg-if/ontology/agent/>), data source (<https://w3id.org/skg-if/ontology/data-source/>), grant (<https://w3id.org/skg-if/ontology/grant/>), research product (<https://w3id.org/skg-if/ontology/research-product/>), topic (<https://w3id.org/skg-if/ontology/topic/>), and venue (<https://w3id.org/skg-if/ontology/venue/>) [16]. Each module is documented on a separate page, and users navigate the ontology network via the module URIs provided among the ontology annotations ("Imported Ontologies" in the left panel of Figure 3).

LODE supports both distributed and aggregated renderings of documentation. Similarly to WIDOCO, imported ontologies can be documented independently (as in https://lode.opencitinations.net/extract?url=https://w3id.org/skg-if/ontology/&read_as=owl). Alternatively, the `imported` parameter resolves the direct imports and generates a single documentation

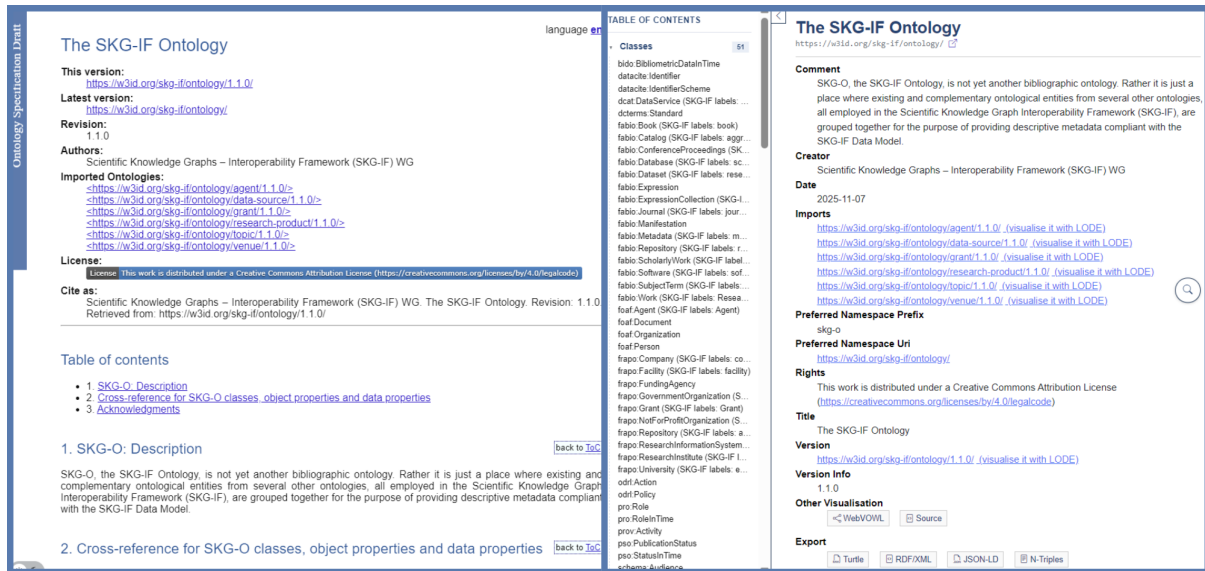


Figure 3: SKG-O ontology documentation generated by WIDOCO (left) and by LOD (right)

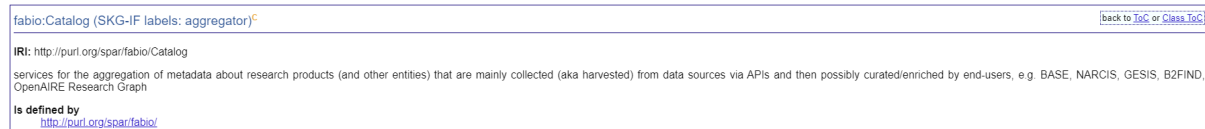


Figure 4: fabio:Catalog (SKG-IF labels: aggregator) class as displayed in SKG-O documentation generated by WIDOCO.

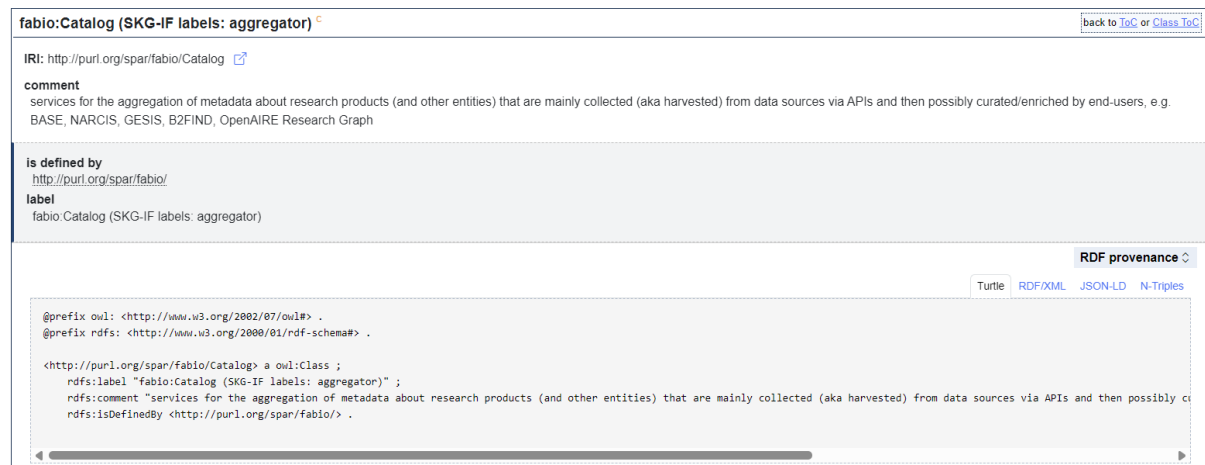


Figure 5: fabio:Catalog (SKG-IF labels: aggregator) class as displayed in SKG-O documentation generated by LOD.

page containing approximately 180 entities, both locally defined and imported (as in https://lode.opencitations.net/extract?url=https://w3id.org/skg-if/ontology/&read_as=owl&imported=true). Given such an extended documentation, simplified access to documented entities is provided through the sidebar and, additionally, the search bar shown in Figure 3.

For what concerns entity documentation, we compare the class *fabio:Catalog* (SKG-IF labels: aggregator) as displayed by WIDOCO (Figure 4) and LOD (Figure 5).

In WIDOCO, the class documentation is displayed with the corresponding label as its title (retrieved by `rdfs:label`), its IRI, its defining ontology (retrieved by `rdfs:isDefinedBy`) and its comment

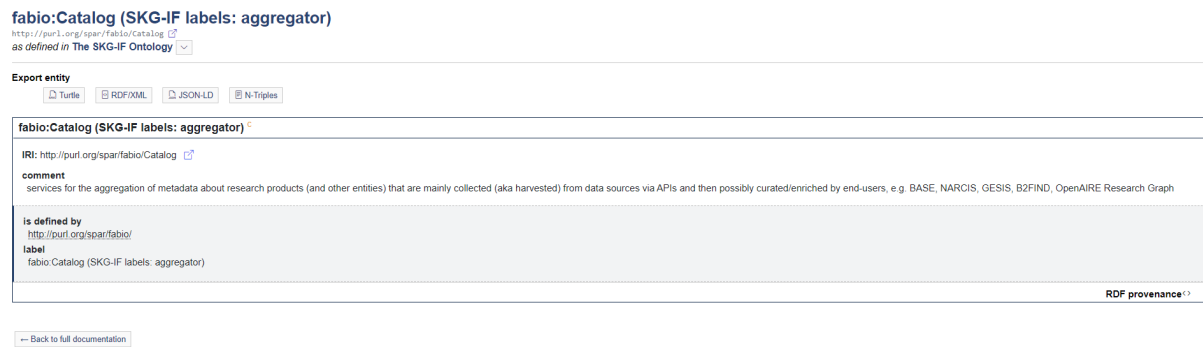


Figure 6: fabio:Catalog (SKG-IF labels: aggregator) class as displayed in SKG-O entity stand-alone documentation generated by LODE.

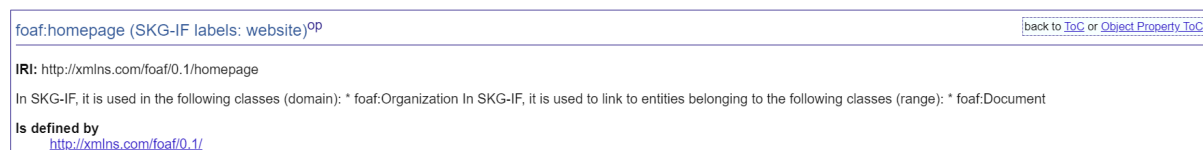


Figure 7: foaf : homepage (SKG-IF label: website) object property as displayed in the SKG-O documentation generated by WIDOCO with imported ontologies enabled.

(retrieved by `rdfs:comment`). LODE’s generated documentation preserves the same information while enriching each entity description with additional documentation-oriented features. These include explicit label metadata (derived from `rdfs:label` to support multiple labels and multilingual descriptions), an *RDF Provenance* panel exposing the RDF statements used to generate the documentation, with direct traceability to the RDF source, and a direct link (external-link icon next to the entity IRI) to the stand-alone documentation page of the entity.

As shown in Figure 6, the stand-alone documentation page of the entity (https://lode.opencitations.net/extract?url=https://w3id.org/skg-if/ontology/&read_as=owl&imported=true&resource=http://purl.org/spar/fabio/Catalog) is documented with the complete set of information previously described, while the page additionally provides contextual metadata about the origin of the documented artefact, including the source ontology (*as defined in the SKG-IF ontology*) and its associated metadata, accessible through a dedicated accordion section. Moreover, the documented entity can be retrieved in four RDF serialisations (Turtle, RDF/XML, JSON-LD, and N-Triples) through the dedicated export functionality (*Export entity*).

Additionally, most SKG-O terms are characterised by pervasive reuse: entities are frequently borrowed from and re-defined across established vocabularies [16] (e.g. several SPAR ontologies [9]). As a consequence, the same entity may be referenced by multiple SKG-O modules, with each module contributing additional annotation assertions that progressively enrich its description. For example, the object property `foaf:homepage` is reused across different modules (e.g. `agent` and `grant`), resulting in a distributed set of annotations associated with the same RDF resource.

Both WIDOCO and LODE support the inclusion of imported ontologies during documentation generation (through WIDOCO’s `-displayDirectImportsOnly` execution option and LODE’s `imported` parameter). Figures 7 and 8 illustrate the documentation of `foaf:homepage` as generated by WIDOCO and LODE with imported ontologies enabled.

WIDOCO-generated documentation displays the object property with its title (retrieved from `rdfs:label`), its IRI, its defining ontology (retrieved through `rdfs:isDefinedBy`), and a textual description. In the latter case, the description is retrieved from `dc:description`, one of the annotation properties explicitly supported by WIDOCO’s documentation mapping. The value contains Markdown syntax, which is displayed verbatim as plain text in the generated documentation. Although

foaf:homepage (SKG-IF labels: website) ^{OP} [back to *IsC* or *Object Property IsC*](#)

IRI: <http://xmlns.com/foaf/0.1/homepage> [↗](#)

domain

thing[?]

range

thing[?]

is defined by

<http://xmlns.com/foaf/0.1/>

label

foaf:homepage (SKG-IF labels: website)

description

In SKG-IF, it is used in the following classes (domain):

- foaf:Organization

In SKG-IF, it is used to link to entities belonging to the following classes (range):

- foaf:Document

In SKG-IF, it is used in the following classes (domain):

- frapo:Grant

In SKG-IF, it is used to link to entities belonging to the following classes (range):

- foaf:Document

RDF provenance [⌵]

Turtle

RDF/XML

JSON-LD

N-Triples

```

@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

foaf:homepage a owl:ObjectProperty ;
  rdfs:label "foaf:homepage (SKG-IF labels: website)" ;
  dc:description ""In SKG-IF, it is used in the following classes (domain):
* foaf:Organization

In SKG-IF, it is used to link to entities belonging to the following classes (range):
* foaf:Document"" ;
  ""In SKG-IF, it is used in the following classes (domain):
* frapo:Grant

In SKG-IF, it is used to link to entities belonging to the following classes (range):
* foaf:Document"" ;
  rdfs:isDefinedBy foaf: .

```

Figure 8: foaf : homepage (SKG-IF label: website) object property as displayed in the SKG-O documentation generated by LODE with imported ontologies enabled.

foaf : homepage is annotated with different descriptions across multiple SKG-O modules, the generated documentation presents a single description for the imported property.

LODE preserves the information displayed by WIDOCO and enriches the entity documentation with the additional documentation-oriented features, as already described while illustrating the case of the fabio : Catalog class shown in Figure 5.

Additionally, the displayed descriptions include annotations from different ontology modules, which are aggregated into a single entity view through cross-module annotation resolution. LODE does not rely on a predefined annotation vocabulary for textual descriptions; therefore, properties such as dc : description, when not explicitly mapped as documentation fields, are preserved and displayed as general statements (modelled as Statement in LODE’s internal model) associated with the entity. This annotation-agnostic approach allows the documentation to expose textual information regardless of the vocabulary used to provide it, supporting future vocabulary changes or newly introduced annotation predicates. Moreover, annotation values are rendered as Markdown whenever applicable, including descriptive literals, preserving the intended formatting of textual documentation. Finally, the entity card is enriched with domain and range information. When explicit axioms are not available, LODE applies documentation-oriented defaults and displays owl : Thing as the inferred domain and range. To preserve transparency between documentation-oriented inferences and axioms explicitly asserted in the ontology, the *RDF Provenance* panel only reports source-level RDF statements; consequently, inferred default values (such as the owl : Thing domain and range shown in this case) are not included among the provenance triples.

6. Conclusions

This work presented the new version of LODE, a re-engineered and extensible framework for generating human-readable documentation of semantic artefacts. LODE reimplements the functionalities provided by its predecessor, including single-page W3C Recommendation-style OWL documentation, imported ontologies, and transitive-closure handling. Additionally, it replaces the original XSLT pipeline with a modular Python-based architecture built upon a strict separation of concerns among its three main components (Reader, Model, and Viewer). Currently, OWL is the first fully supported semantic artefact family; however, the decoupled architecture is designed to accommodate additional semantic resource types, such as RDF(S) vocabularies, SKOS concept schemes, SHACL shapes, and other knowledge representation resources, through dedicated extensions.

The current implementation provides full support for OWL artefacts and introduces several improvements aimed at addressing the documentation requirements of modern semantic models. In particular, LODE extends the original documentation workflow by improving (i) documentation generation and completeness (in particular, by providing documentation-oriented inferences and default values and rendering rich-text annotations), (ii) entity-level documentation and traceability (in particular, by adding annotation-agnostic statements, per-entity RDF provenance, and entity stand-alone documentation), (iii) navigation and discovery of documented entities (in particular, by providing a persistent, collapsible sidebar for ontology structural navigation aid and a client-side search bar), and (iv) the generation and deployment of documentation artefacts (in particular, by providing local artefact upload and static documentation generation through the `/build` endpoint).

The applicability of these features has been demonstrated by comparing the current SKG-O documentation generated with WIDOCO and the documentation produced by the new version of LODE for the same ontology. The comparison highlights differences in annotation coverage, rendering strategies, and newly introduced documentation capabilities. It demonstrates that LODE provides richer documentation outputs by aggregating entity descriptions distributed across imported ontology modules, preserving heterogeneous annotations independently of the adopted vocabulary, rendering rich-text documentation consistently, and exposing the RDF provenance of each documented entity.

Currently, this new version of LODE is being used to develop the new documentation pages for all the SPAR Ontologies, largely extending their annotations to provide even more self-contained documentation. Similarly, we plan to use LODE to provide the new documentation for SKG-O, thereby making the SKG Data Model clearer for users.

Additional future work will focus on extending the range of supported semantic artefact families while reusing the existing modular architecture. The first planned extensions concern the introduction of dedicated modules for RDF resources without ontology-level semantics and for SKOS concept schemes, followed by support for SHACL shapes. Moreover, future developments will investigate interactions among different semantic artefact modules, such as the combined documentation of OWL ontologies and SHACL shapes. We will also work on enabling LODE to use an OWL reasoner to complement its current lightweight, documentation-oriented inference mechanisms.

Finally, LODE's modular, decoupled architecture has been designed to support its integration into LUMIS (the LUMEN Infrastructure for Semantics)⁴⁵, a collaborative platform that supports the lifecycle of FAIR-by-design semantic artefacts, from requirement definition and editing to publication and maintenance. Currently disseminated as a stand-alone tool, LODE is intended to be integrated into LUMIS to provide human-readable documentation for these artefacts.

Acknowledgments. This work has been funded by the European Union's Horizon Europe framework programme under Grant Agreement No 101187940 (LUMEN).

⁴<https://lumenproject.eu/lumis-lumen-infrastructure-for-semantics/>

⁵LUMEN (Linked User-driven Multidisciplinary Exploration Network) is a Horizon Europe project of the European Open Science Cloud (EOSC), aimed at fostering interoperability and cross-domain discovery across Mathematics, Social Sciences and Humanities, Earth System, and Molecular Dynamics. LUMEN aims at providing researchers with AI-powered tools supporting the early phases of the research process. See <https://eosc.eu/horizon-europe-projects/eosc-lumen>.

Declaration of Use of Generative AI. No generative AI systems were used to generate or analyse data, or draft the scientific content of this manuscript. AI-assisted tools were used only for spell-checking and minor grammar suggestions on the English text. Generative AI-based coding assistants were used during the development of LODE to support code generation and, especially, review; all AI-suggested code was reviewed, tested, and validated by the authors.

References

- [1] D. Shotton, K. Portwin, G. Klyne, A. Miles, Adventures in Semantic Publishing: Exemplar Semantic Enhancements of a Research Article, *PLoS Computational Biology* 5 (2009) e1000361. doi:10.1371/journal.pcbi.1000361.
- [2] D. Shotton, Semantic publishing: the coming revolution in scientific journal publishing, *Learned Publishing* 22 (2009) 85–94. doi:10.1087/2009202.
- [3] A. García Castro, C. Lange, P. Lord, R. Stevens (Eds.), *Semantic Publishing*, volume 1155 of *CEUR Workshop Proceedings*, CEUR-WS, Aachen, Germany, 2014. URL: <https://ceur-ws.org/Vol-1155/>.
- [4] A. González-Beltrán, F. Osborne, S. Peroni, S. Vahdati (Eds.), *Semantics, Analytics, Visualization*, volume 10959 of *Lecture Notes in Computer Science*, Springer, Cham, Switzerland, 2018. doi:10.1007/978-3-030-01379-0.
- [5] A. Jacyszyn, A. Mannocci, F. Osborne, G. Rehm, A. Salatino, S. Schimmler, L. Stork (Eds.), *Scientific Knowledge: Representation, Discovery, and Assessment 2025*, volume 4065 of *CEUR Workshop Proceedings*, CEUR-WS, Aachen, Germany, 2025. URL: <https://ceur-ws.org/Vol-4065/>.
- [6] O. Corcho, K. Kurowski, M. Ojsteršek, C. Choirat, M. van de Sanden, F. Coppens, EOSC interoperability framework, Technical Report, Publications Office of the European Union, Luxembourg City, Luxembourg, 2021. URL: <https://doi.org/10.2777/620649>.
- [7] W. Nyberg Åkerström, K. Baumann, O. Corcho, R. David, Y. Le Franc, B. Madon, B. Magagna, A. Micsik, M. Molinaro, M. Ojsteršek, S. Peroni, A. Scharnhorst, L. Vogt, H. Widmann, Developing and implementing the semantic interoperability recommendations of the EOSC Interoperability Framework, Deliverable, EOSC Association, 2024. URL: <https://doi.org/10.5281/zenodo.10843882>.
- [8] K. Baumann, B. Cecconi, M. Dietrich, M. Doerr, T. Jouneau, A. Kokkinaki, J. Marek, M. Med, P. Tewatia, Landscape overview for the EOSC Federation Semantic Interoperability in 2025, Deliverable, EOSC Association, Brussels, Belgium, 2026. doi:10.5281/zenodo.19134873.
- [9] S. Peroni, D. Shotton, The SPAR Ontologies, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. M. Zurada (Eds.), *The Semantic Web – ISWC 2018*, volume 10842 of *Lecture Notes in Computer Science*, Springer, Cham, Switzerland, 2018, pp. 119–136. doi:10.1007/978-3-030-00668-6_8.
- [10] S. Peroni, D. Shotton, OpenCitations, an infrastructure organization for open scholarship, *Quantitative Science Studies* 1 (2020) 428–444. doi:10.1162/qss_a_00023.
- [11] S. Auer, A. Oelen, M. Haris, M. Stocker, J. D’Souza, K. E. Farfar, L. Vogt, M. Prinz, V. Wiens, M. Y. Jaradeh, Improving Access to Scientific Literature with Knowledge Graphs, *Bibliothek Forschung und Praxis* 44 (2020) 516–529. doi:10.1515/bfp-2020-2042.
- [12] V. A. Carriero, M. Daquino, A. Gangemi, A. G. Nuzzolese, S. Peroni, V. Presutti, F. Tomasi, The Landscape of Ontology Reuse Approaches, in: G. Cota, M. Daquino, G. L. Pozzato (Eds.), *Applications and Practices in Ontology Design, Extraction, and Reasoning*, volume 49 of *Studies on the Semantic Web*, IOS Press, Amsterdam, The Netherlands, 2020, pp. 21–38. URL: <https://arxiv.org/abs/2011.12599>. doi:10.3233/SSW200033.
- [13] S. Peroni, D. M. Shotton, F. Vitali, The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation, in: A. Teije, J. Völker, S. Handschuh, H. Stuckenschmidt, M. d’Aquin, A. Nikolov, N. Aussenac-Gilles, N. Hernandez (Eds.), *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012*, Galway City, Ireland, October 8–12, 2012. Proceedings, volume 7603 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, 2012, pp. 398–412. doi:10.1007/978-3-642-33876-2_35.
- [14] S. Peroni, D. M. Shotton, F. Vitali, Tools for the Automatic Generation of Ontology Documentation: A Task-Based Evaluation, *International Journal on Semantic Web and Information Systems* 9 (2013) 21–44. doi:10.4018/jswis.2013010102.
- [15] D. Garijo, WIDOCO: A Wizard for Documenting Ontologies, in: C. d’Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, J. Heflin (Eds.), *The Semantic Web – ISWC 2017*, Springer International Publishing, Cham, 2017, pp. 94–102. doi:10.1007/978-3-319-68204-4_9.

- [16] A. Mannocci, S. Peroni, RDA Scientific Knowledge Graphs Interoperability Framework (SKG-IF) WG, Scientific Knowledge Graphs Interoperability Framework (SKG-IF): Enabling Data Exchange for SKGs, 2026. doi:10.15497/RDA00150.
- [17] V. Pasqual, S. Peroni, A. Nazari, LOD - Live OWL Documentation Environment, 2026. doi:10.5281/zenodo.21805185.
- [18] S. Peroni, L. Asprino, L. Prete, A. Russo, A. Serafini, <https://github.com/essepuntato/LODE>, 2020. URL: <https://archive.softwareheritage.org/swh:1:snp:eae25a71df8e73bdc4ec3d8d8866293b6e1bfea7;origin=https://github.com/essepuntato/LODE>.
- [19] I. Palmisano, S. Spero, P. Ansell, M. Horridge, <https://github.com/owlcs/owlapi>, 2024. URL: <https://archive.softwareheritage.org/swh:1:snp:732f5023b8b3798655a50f57aac717d04577b4f3;origin=https://github.com/owlcs/owlapi>.
- [20] M. Kay, XSL Transformations (XSLT) Version 2.0 (Second Edition), 2021. URL: <https://www.w3.org/TR/xslt20/>.
- [21] D. Garijo, WIDOCO: A wizard for documenting ontologies, 2024. URL: <https://archive.softwareheritage.org/swh:1:dir:40d462bbecefc3a9c3e810567d1f0d7606e0fae7;origin=https://github.com/dgarijo/Widoco;visit=swh:1:snp:4d8570d390a18fee1d37f262ad79b36c021c611a;anchor=swh:1:rev:fec66b89a4f4acb015a44cf8cb671d49bec626a>.
- [22] N. Car, <https://github.com/RDFLib/pyLODE>, 2026. URL: <https://archive.softwareheritage.org/swh:1:snp:bf2e380cac3da07655d3f7bb0c910cf9876c64cd;origin=https://github.com/RDFLib/pyLODE>.
- [23] D. Krech, G. A. Grimnes, G. Higgins, J. Hees, I. Aucamp, N. Lindström, N. Arndt, A. Sommer, E. Chuc, I. Herman, A. Nelson, J. McCusker, T. Gillespie, T. Kluyver, F. Ludwig, P.-A. Champin, M. Watts, U. Holzer, E. Summers, W. Morris, D. Winston, D. Perttula, F. Kovacevic, R. Chateaneu, H. Solbrig, B. Cogrel, V. Stuart, RDFLib, 2026. doi:10.5281/zenodo.6845245, version 7.4.0.
- [24] S. Lohmann, V. Link, E. Marbach, S. Negru, WebVOWL: Web-based Visualization of Ontologies, in: P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, C. Ghidini (Eds.), Knowledge Engineering and Knowledge Management, Springer International Publishing, Cham, 2015, pp. 154–158. doi:10.1007/978-3-319-17966-7_21.
- [25] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, S. Rudolph, OWL 2 Web Ontology Language Primer (Second Edition), 2012. URL: <https://www.w3.org/TR/owl2-primer/>.
- [26] C. Jeffrey, Marked js, a markdown parser and compiler. built for speed., 2026. URL: <https://archive.softwareheritage.org/swh:1:rel:5c5a5be49c166d1974a9a6e7254ae8376bc85cef;origin=https://www.npmjs.com/package/marked;visit=swh:1:snp:55f66de754fc7ef0b19e119bf2b130b32a613bb3,version 18.0.12>.
- [27] M. Heiderich, M. Kinugawa, Dompurify - a dom-only, super-fast, uber-tolerant xss sanitizer for html, mathml and svg, 2020. URL: <https://archive.softwareheritage.org/swh:1:rel:3ecad9dc69ea01eae4fb24238d2ffbe4458e5cc6;origin=https://www.npmjs.com/package/dompurify;visit=swh:1:snp:cecc5c8f764c3ee0a5e7125499b4f1a8bfc591cf,version 3.4.15>.
- [28] A. Ronacher, D. Lord, Jinja: A very fast and expressive template engine, 2024. URL: <https://archive.softwareheritage.org/swh:1:rel:7f9526900a4df6a70a527ce7af4ec41adeedd783;origin=https://github.com/pallets/jinja;visit=swh:1:snp:175a4898b890b21ac7d3a16993f47fd8f0c59278,version 3.2.1>.
- [29] J. Déramond, C. Oliff, Bootstrap. sleek, intuitive, and powerful front-end framework for faster and easier web development., 2024. URL: <https://archive.softwareheritage.org/swh:1:dir:07a9b3d274133f5cfc2681459b7bab2000df8b98;origin=https://github.com/twbs/bootstrap;visit=swh:1:snp:3e8c14f504bd12eb3ec45c817df7fa8bf85ef3bf;anchor=swh:1:rev:d50c38fa2cf65d05b3dd65853adf2aa056d93031,version 5.3.3>.