Knowledge graph construction for the Herstory project

Version 1: 20250313

Authors: Miquel Centelles, Juan Antonio Pastor, Marina Salse, Tomás Saorín, Núria Ferran-Ferrer

Content

Preliminary process

- Determination of the schedule, roles, and outcomes.
- Coordination with the contracting company and doctoral and master's students.

Phase 1: Preparation of Data Sources

Reference methodology: Radulovic et al. (2015)

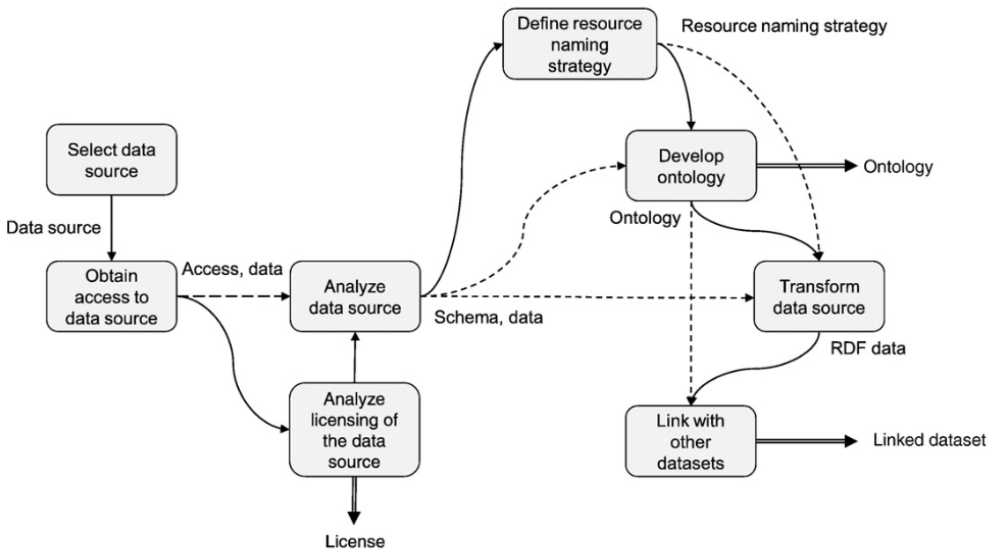


Ilustración 1: Radulovic et al., 2015, p. 180

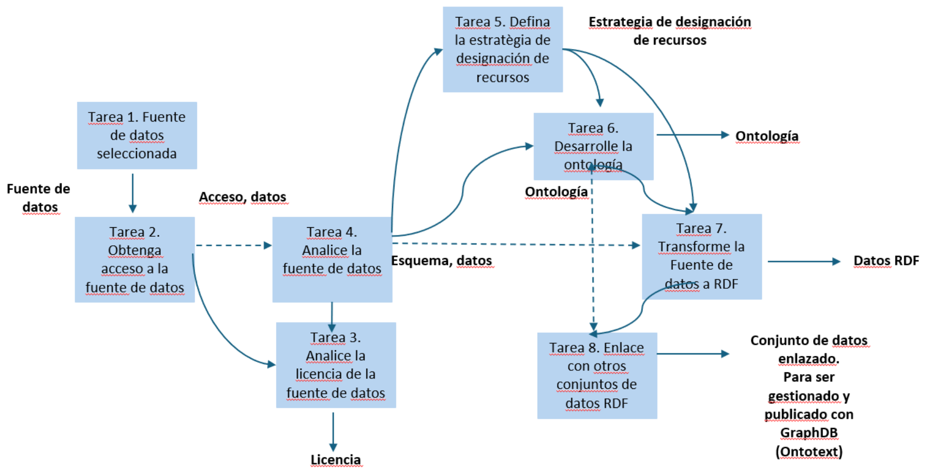

Tarea 1. Fuente de datos seleccionada
Fuente de datos
Tarea 2. Obtenga acceso a la fuente de datos
Acceso, datos
Tarea 4. Analice la fuente de datos
Esquema, datos
Tarea 3. Analice la licencia de la fuente de datos
Licencia
Tarea 5. Defina la estratègia de designación de recursos
Estrategia de designación de recursos
Tarea 6. Desarrolle la ontología
Ontología
Ontología
Tarea 7. Transforme la Fuente de datos a RDF
Datos RDF
Tarea 8. Enlace con otros conjuntos de datos RDF
Conjunto de datos enlazado. Para ser gestionado y publicado con GraphDB (Ontotext)

Ilustración 2: Radulovic et al., 2015, p. 180

Process 1. Obtaining access to the source databases

1.1. Legal aspects

- Negotiation of usage and transformation licenses for the database.
- Formalization of agreements on usage and transformation licenses for the database.
- Documentation of the conditions of the usage and transformation licenses for the database.

Output: Licenses.

1.2. Technical aspects

- Determination of the modality for the acquisition of the database files.
- Determination, obtaining, and implementation of the necessary software and hardware resources for the use of the databases.

Output: Access to the databases.

Process 2. Analysis of the source databases

- Identification of data schemas through:
    - Existing documentation (especially, logical design).
    - Direct inspection.
- Identification of data dictionaries through:
    - Existing documentation.
    - Direct inspection.

Output: Comprehension about data schemas and data dictionaries of the databases.

Process 3. Definition of the resource designation strategy (IRI/URI)

- Resource designation strategy (IRI/URI) for ontology components.
- Resource designation strategy (IRI/URI) for individual resources (instances of classes).
- Management, provisioning, and dereferencing model for URI/IRI.

Output: Resource designation strategy protocol.

Phase 2. Development of the Ontology

Reference methodologies:

- ''Ontology Development 101: A Guide to Creating Your First Ontology'' (Noy & McGuinness, 2001)

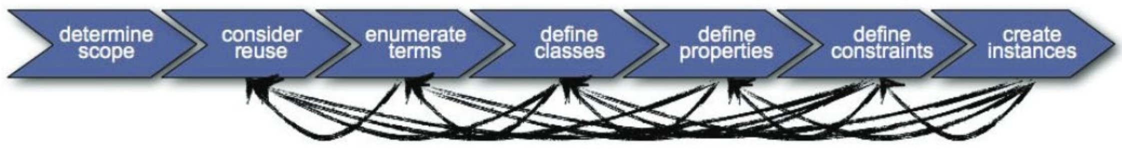


Illustration 3: Sack & Alam, 2020, p. 16

- Linked Open Terms (LOT) methodology: (Poveda-Villalón et al., 2022a)

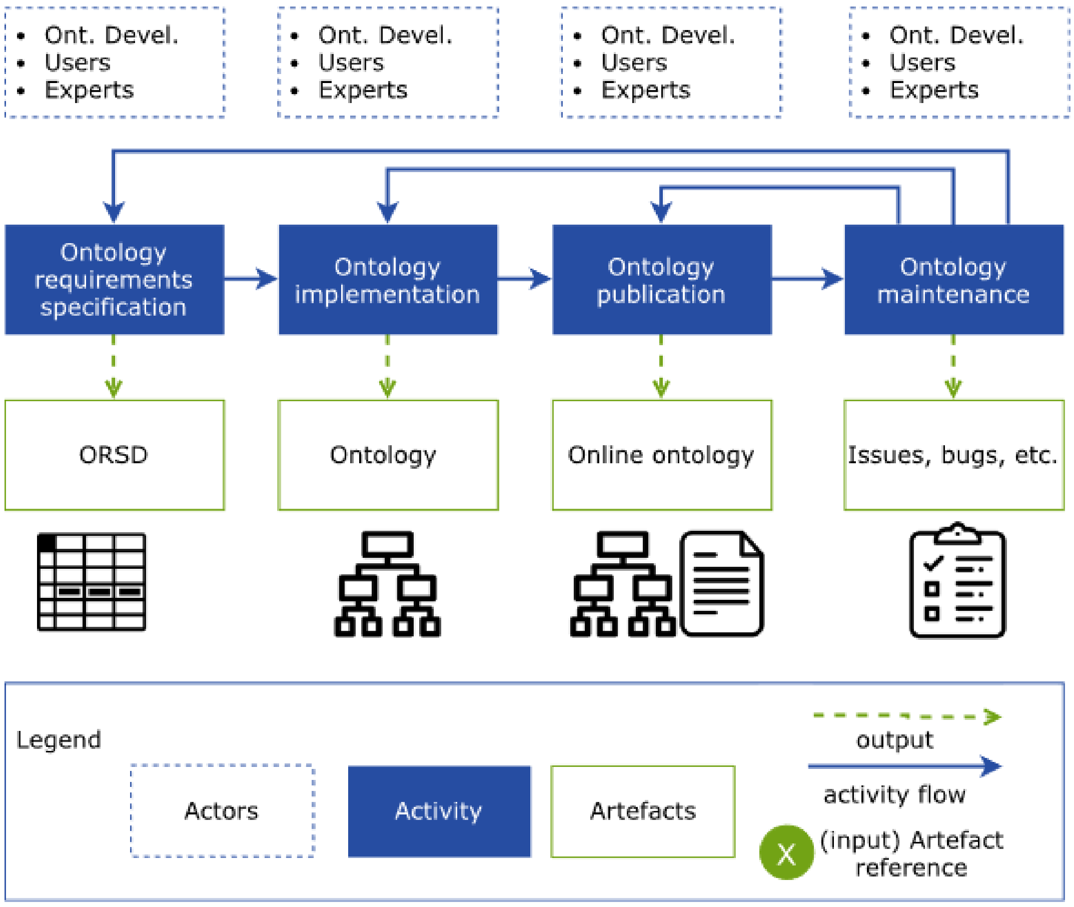


Illustration 4: Poveda-Villalón et al., 2022, p. 3

Process 1: Selection of the reference ontology/ontologies

Objective: Identify and select one or more ontologies for reuse in the project. Ontology reuse refers to the process of using available ontological resources for solving different problems (Suárez-Figueroa et al., 2015). The ontology reuse process may be considered along the following processes:

1. Ontology requirement specification. When the development team could define the conditions that an ontology should satisfy to be reused.
2. Ontology implementation. It is the process where reuse actually takes place. The ontology conceptualization might help to identify what entities could be reused. For example, during conceptualization, the necessity of modeling the moment when a sensor observation appears leads to consider the OWL Time ontology reuse. During the encoding process, ontologies are soft or hard reused. During evaluation, the appropriate use of reused terms according to their intended and formal meaning as well as the coherence of the whole ontology are checked.
3. Ontology maintenance. The future evolution of the ontology may lead to the reuse of new ontologies or even to discontinue reusing ontologies integrated in former versions.

Subprocesses:

- Comparison with similar contexts based on the scoping review.
- Thorough search in ontology repositories and registries.

Resources:

- Main resource: *Linked Open Vocabularies (LOV)* (2025)
- Complementary resources:
  - *Archivo: Ontology Archive* (2025)
  - *BARTOC.org* (2020)
  - *CommonCoreOntology/CommonCoreOntologies* (2017/2025)
  - *OntoHub* (2025)
  - *re3data.org* (2013)

Output: One or more reference ontologies selected for reuse in the project.

Process 2: Ontology requirements specification

Objective: The ontology requirements specification process refers to the process of collecting the requirements that the ontology should fulfill (Suárez-Figueroa et al., 2015).

This process includes:

- Analysis and discussion of the project's general objectives.
- In-depth review of the literature (scoping review).
- Identification and comparison of similar cases based on the in-depth literature review.

Output: This process results the ontology requirements specification document (ORSD) (Suárez-Figueroa et al., 2009), which gathers the information related to the requirements, including the use case specification, the documentation needed, the purpose and scope of the ontology and the requirements proposal.

Subprocesses, according to Poveda-Villalón et al. (2022b)

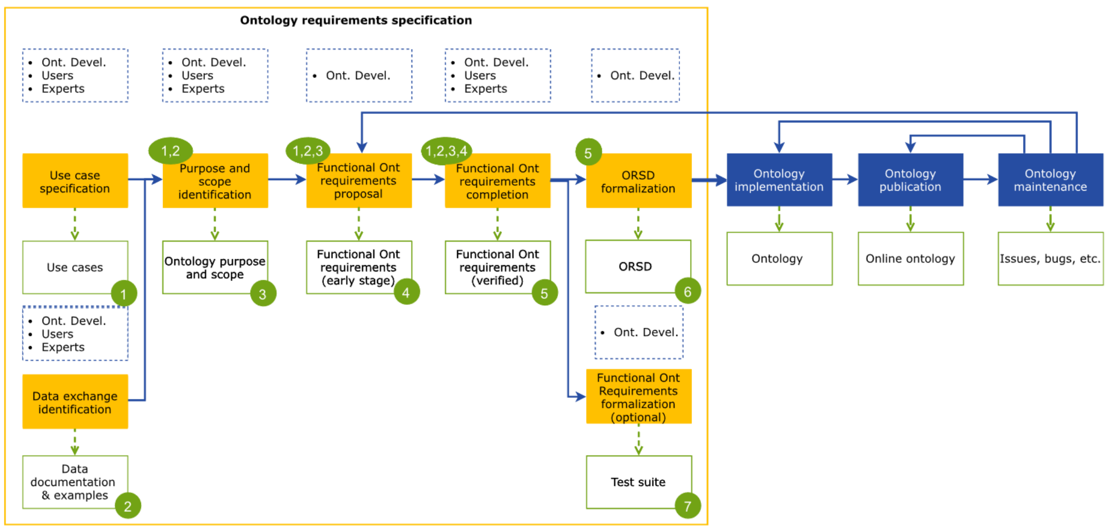

Ontology requirements specification
Ont. Devel. Users Experts
Ont. Devel. Users Experts
Ont. Devel.
Ont. Devel. Users Experts
Ont. Devel.
Use case specification
1,2
Purpose and scope identification
1,2,3
Functional Ont requirements proposal
1,2,3,4
Functional Ont requirements completion
5
ORSD formalization
Ontology implementation
Ontology publication
Ontology maintenance
Use cases
1
Ontology purpose and scope
3
Functional Ont requirements (early stage)
4
Functional Ont requirements (verified)
5
ORSD
6
Ontology
Online ontology
Issues, bugs, etc.
Ont. Devel. Users Experts
Ont. Devel.
Data exchange identification
Functional Ont Requirements formalization (optional)
Data documentation & examples
2
Test suite
7

Illustration 5: Poveda-Villalón et al., 2022, p. 5

2.1. Use case specification

Objective: The goal of the use case specification process is to provide a vision on the potential use that the ontology will have.

Output: a list of use cases.

2.2. Data exchange identification

Objective: The goal of the data exchange identification process is to provide the ontology development team with the necessary documentation about the domain to be modeled.

Output: a set of domain documents and resources.

2.3. Purpose and scope identification

Output: As output of this process a text document describing the purpose and scope of the ontology is produced. This document could include nonfunctional requirements as the language in which the ontology should be lexicalized, whether it has to provide multilingual lexicalizations, the ontology implementation language or profile in which it should be formalized, etc.

2.4. Functional ontological requirements proposal

Output: This process is expected to produce a set of functional requirements that could be materialized into one or more of the following forms: Competency questions (CQs); Natural language statements; and Tabular information including concepts, relations, and attributes.

Specific resource: CORAL Corpus (Fernández-Izquierdo et al., 2017)

2.5. Functional ontological requirements completion

Objective: Domain experts and users in collaboration with the ontology development team validate whether the ontology requirements defined in the previous step are valid and complete.

Specific resource: CORAL Corpus (Fernández-Izquierdo et al., 2017)

2.6. ORSD formalization

Objective: This specification document stores all the functional and non-functional requirements identified and the information associated to them.

Specific resource: Templates for ORSD (*oeg-upm/LOT-resources*, 2020/2025)

2.7. Functional ontology requirements formalization

Output: In this (optional) task, the functional requirements are formalized into test cases. These tests cases should include: the identifier of the requirement associated, the description of the test case (which includes a link to the ORSD), and the SPARQL queries extracted from the CQ together with the expected result of the query.

Specific Resource: THEMIS (Fernández Izquierdo, 2020)

Process 3: Ontology implementation

Objective: The aim of the ontology implementation process is to build the ontology using a formal language, based on the ontological requirements identified by the domain experts and the ontology development team (Suárez-Figueroa et al., 2015).

The ontology development team builds the ontology iteratively, implementing only a certain number of requirements in each iteration. The output of each iteration is a new version of the ontology.

Output: Ontology.

Subprocesses, according to Poveda-Villalón et al. (2022b):

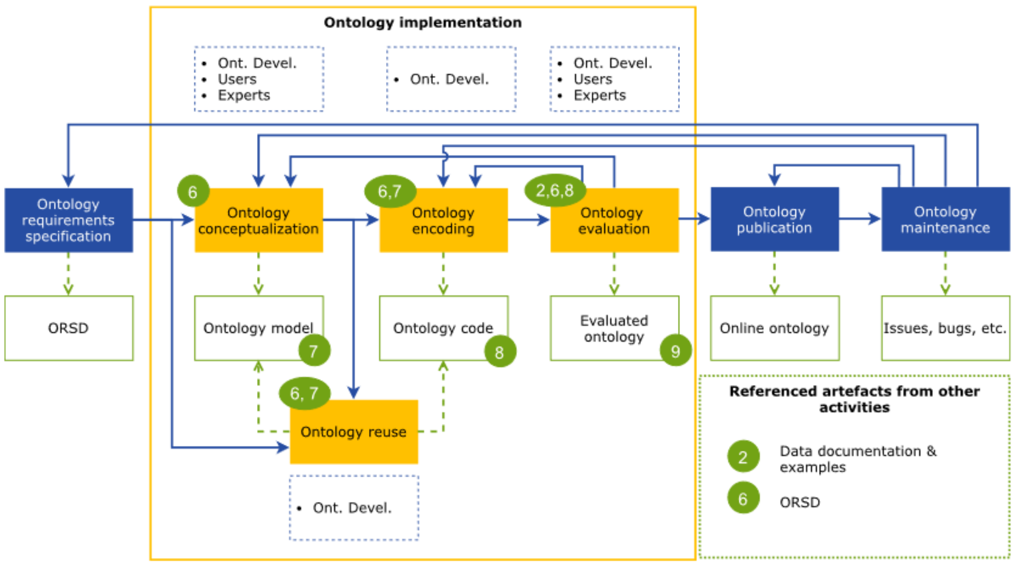


Illustration 6: Poveda-Villalón et al., 2022, p. 8

3.1. Conceptualization through the ontological transformation of the schema and data dictionary of the five databases

The purpose of the conceptualization subprocess is to create a model that represents the domain of the ontology. Such conceptualization usually does not include all the constraints that a formal language imposes; instead the level of detail of is decided by the development team. For example, if the conceptualization is going to be used for communication purposes with domain experts, some language specific constraints might be omitted, generating a more high level model.

This process includes.

- Ontological transformation of each of the databases individually, which includes:
  - Generation of classes, including the class hierarchy. For each class, one or more labels will be established (in Catalan, Spanish, and English), along with definitions by intension and extension.

    - Generation of properties (*owl:ObjectProperty* and *owl:DatatypeProperty*), including the property hierarchy. For each class, one or more labels will be established (in Catalan, Spanish, and English), along with definitions by intension and extension.
    - Generation of restrictions and axioms for classes and properties (for example: cardinality constraints, property characteristics, universal or existential constraints, etc.)
- Harmonization and integration of the results from the ontological transformation of the five databases into a single ontology. Application of the reference ontology/ontologies as a tool for harmonization and integration.

Output: Ontology model.

Specific resources:

- Transformation rules:
    - *Reglas básicas de transformación de bases de datos relacionales en bases de datos RDF* [Prepared by the author].
    - Additional bibliography on transformation rules. Collection "Mapping relational database schemas into ontologies" in Zotero [Prepared by the author].
- Transformation tools:
    - *Chowlk* (2024)*:* An on-line application and web service that transforms XML (Extensible Markup Language) ontology diagrams, generated with diagrams.net, into OWL code. The *Chowlk* notation is provided as diagrams.net template to be imported in diagrams.net so that the user could reuse the OWL building blocks as commonly used for UML (Unified Modeling Language) editors.
    - *Gra.fo* (2015): which is an online and collaborative tool for visually designing knowledge graphs. It also allows importing OWL and Turtle file formats, although to the date only classes, subclasses, data properties, and object properties are imported.
    - *Graffoo* (Peroni, 2011): *Graffoo* is an open source tool that can be used to depict the classes, properties and restrictions within OWL ontologies, or sub-sections of them. It defines a set of elements for displaying entities, relationships and restrictions in the diagram.
    - *OWLAx* (Sarker, 2016/2023): There is also a plugin for Protégé, named *OWLAx*, which provides an interface for drawing class diagrams, and a command to generate axioms from the given diagram.
    - *OWLGred* [Not available]: A graphical editor for OWL ontologies that can be used for diagram generation. It offers ontology interoperability (import/export) functionality with *Protégé* and ontology authoring features. The system can be used for online visualization or downloaded for local installation.
    - *WebVOWL* (2025): Originally was designed as an ontology visualization tool following the *WebVOWL* notation and the corresponding ontology editor was provided as separated software. Later, the edition tools were merged within *WebVOWL*. The system could be used online or downloaded for local installation.
    - Diagramming systems: *diagrams.net*; *yEd*; etc.
    - Analogical or digitals blackboards

3.2. Ontology encoding (OWL and RDF(S))

Objective: The goal of the encoding subprocess is to produce an ontology in an implementation language, specifically, RDFS, OWL, and for constructs related to controlled vocabularies such as thesauri, SKOS.
In this subprocess, the activities to be performed are:

- Use the standards of the applied languages (RDFS, OWL, and SKOS), as outlined in W3C (2025).
- Important: the labels of each class and property will be encoded using *rdfs:label* properties; natural language descriptions using *rdfs:comment* properties; and definitions using *rdfs:isDefinitionBy* properties.

Resources:

- Main resources: *VocBench* (2017)
- Complementary resources:
    - *FluentEditor: Ontology Editor* (2025)
    - *PoolParty* (2025)
    - *Protégé* (2025)
    - *TopBraid Composer* (2025)

3.3. Ontology reuse

In this subprocess, the activities to be carried out are:

- Determination of the reuse model between *Hard reuse* and *Soft reuse*. See "3.2.3. Ontology reuse" in Poveda-Villalón et al. (2022b, pp. 9-10).
- Differentiation between resources and attributes available in the five source databases (scenarios) and the resources and attributes available in external data sets.
- In the case of resources and attributes available in external data sets, determination of the data linking model, from the conceptual point of view (expression in the ontology) and from the technological point of view (linking through federated SPARQL searching).

3.4. Ontology evaluation

Objective: This process refers to checking the technical quality of an ontology against a frame of Suárez-Figueroa et al. (2015). Such checking may be carried out considering different evaluation criteria, such as domain coverage, fit for purpose or application, detection of bad practices or logical consistency checking.

Evaluation can be divided into two categories (Suárez-Figueroa et al., 2013):

(1) validation, which compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize; and
(2) verification, which compares the ontology against the ontology specification document (ontology requirements and competency questions), ensuring that the ontology is built correctly (in compliance with the ontology specification requirements collected in the ORSD).

Resources:

- Compilation of ontology evaluation techniques: Centelles y Ferran (2024); Centelles y Ferran (2024).

- Semi-automated evaluation tools.
  - Main resource: reasoners in *Ontotext GraphDB* (2025)
  - Complementary resources:
    - *The OntoCheck Plugin* (2011)
    - *OOPS!: OntOlogy Pitfall Scanner!* (2023)
    - *Tddonto* (2016/2017)
    - *THEMIS* (2020)
    - Other reasoners: *Hermit* (2009); *Pellet* (2013/2025); etc.

Process 4: Ontology publication

Objective: The aim of the ontology publication process is to provide an online ontology accessible both as a human-readable documentation and a machine-readable file from its URI. The ontology publication is usually divided into the following sub-processes as shown in *Illustration 7*.

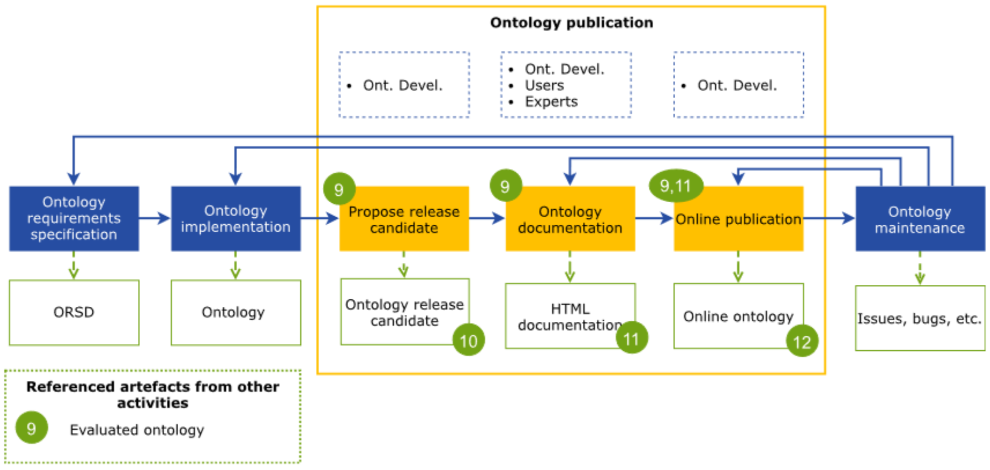


Illustration 7: Poveda-Villalón et al., 2022, p. 11

The ontology development team will determine when it will be published and under what conditions. Specifically, it will establish how the following subprocesses will be carried out.

4.1. Propose release candidate

Objective: Once the ontology developers have implemented and evaluated the ontology, the next task is to decide whether the current version is going to be published on the Web (or shared among other project partners, for example software developers making use of the ontology) or whether is needed to document such version of the ontology for any other reason (for example a set of requirements are fulfilled and triggers a new release).

Output: Ontology release candidate.

4.2. Ontology documentation

Objective: If the ontology has been selected as release, or there are other reasons to document it, then the ontology development team in collaboration with the domain experts generates the ontology documentation taking as input the ontology code and potentially other artifacts as requirements, tests, examples, etc.

Resources:

- *LODE: Live OWL Documentation Environment* (Peroni, 2013)
- *pyLODE* (2024)
- *WIDOCO: A wizard for documenting ontologies* (2013/2017)

Output: HTML documentation

4.3. Online publication

Objective: The ontology is published on the Web following the practices described for publishing vocabularies on the Web provided by the *Best Practice Recipes for Publishing RDF Vocabularies* (W3C, 2008)so that it is accessible via its URI as a file in a formal language and as human-readable documentation using content negotiation.

Resources:

- *OnToology* (2025)
- *Vocol* (2014/2023)
- Web server

Output: Online ontology

Process 5: Ontology maintenance

Objective: The goal of this process is to update the ontology during its life cycle. This may be needed due to different situations as shown in *Illustration 8*. In this case, the internal boxes shown in *Illustration 8* represent potential situations that may occur, and that the ontology development team should react to, rather than processes that the ontology development team should carry out.

Subprocesses, according to Poveda-Villalón et al. (2022b)

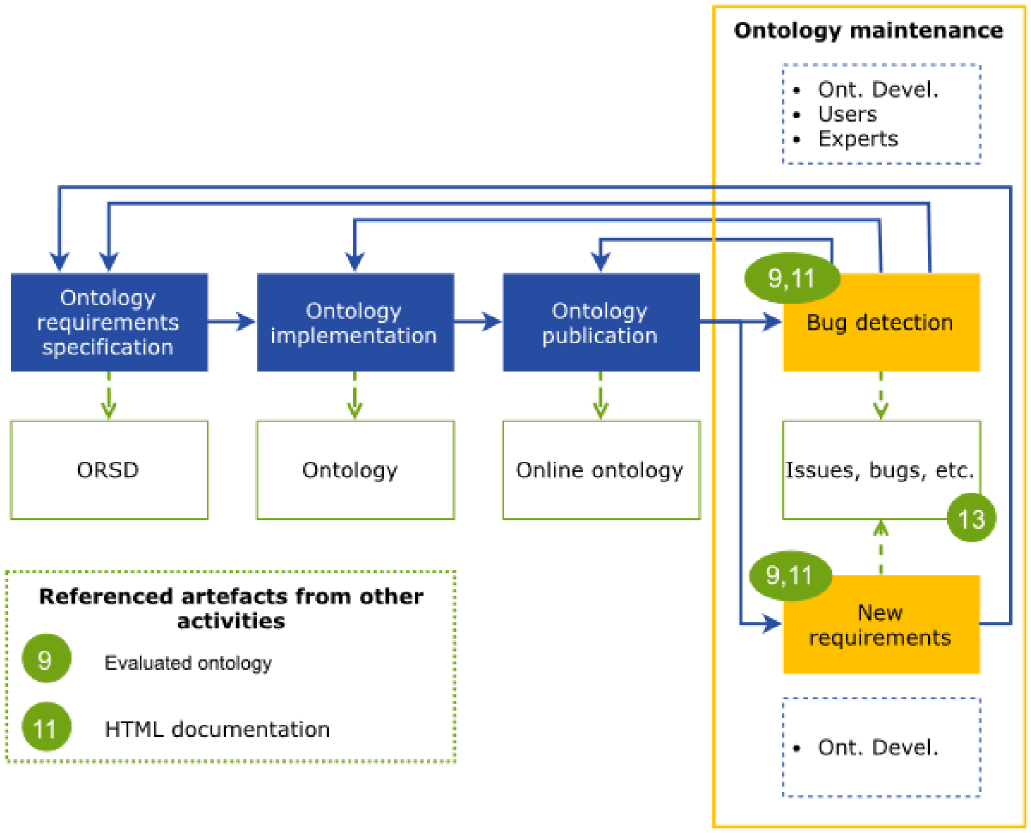


Illustration 8: Poveda-Villalón et al., 2022, p. 12

The subprocesses of Bug detection and new requirements have the identification of issues, bugs, etc. as a output.

Resources:

- *BitBucket* (2025a)
- *GitHub* (2025)
- *GitLab* (2025)
- *Gra.fo* (2015)
- *Jira* (2025b)
- *Mantis* (2025)

Fase 3. Development of the Knowledge Graph

Reference methodologies.

- LOT4KG: LOT for Knowlege Graphs Construction: *LOT4KG: LOT for Knowlege Graphs Construction* (2024); Pernisch et al. (2025)

  This methodology establishes extensions in three areas: Ontology Lifecycle Extension, Knowledge Graph Engineering Extension, and Knowledge Graph Lifecycle Extension. Of the three, only the last two need to be carried out, since the project includes the development of an ontology from its inception. The processes of developing the ontology and the knowledge graph run in parallel.

Knowledge Graph Engineering Extension

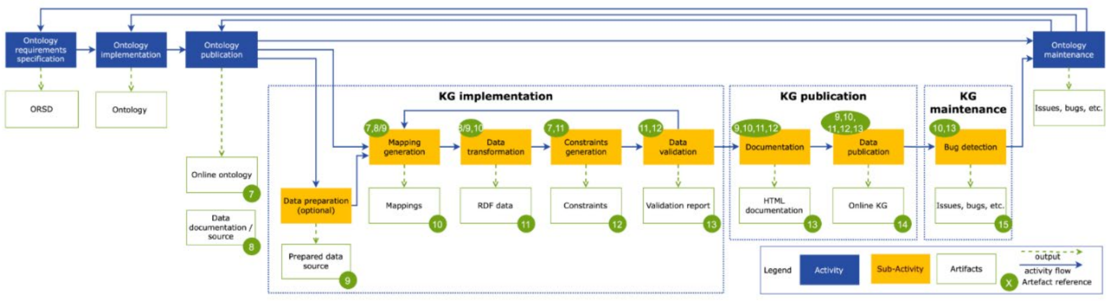


Ilustración 9: LOT4KG: LOT for Knowlege Graphs Construction, 2024

Knowledge Graph Lifecycle Extension

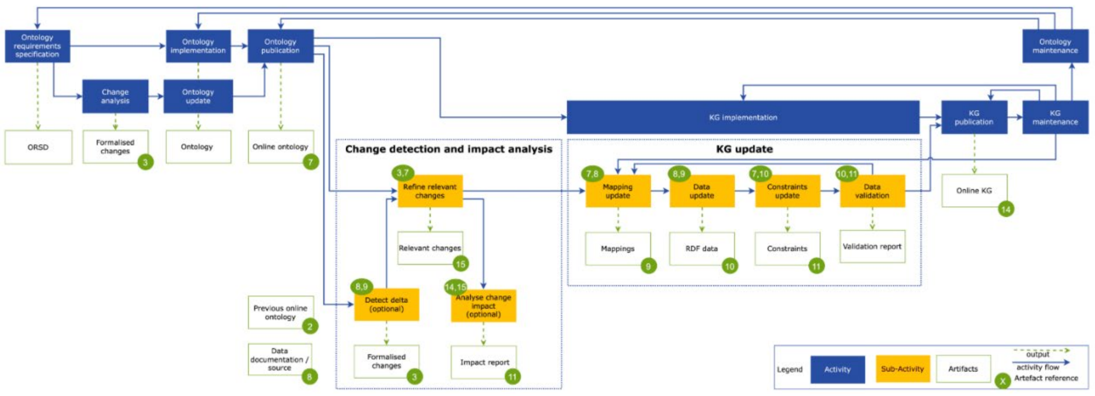


Ilustración 10: LOT4KG: LOT for Knowlege Graphs Construction, 2024

- Simsek et al. (2021)

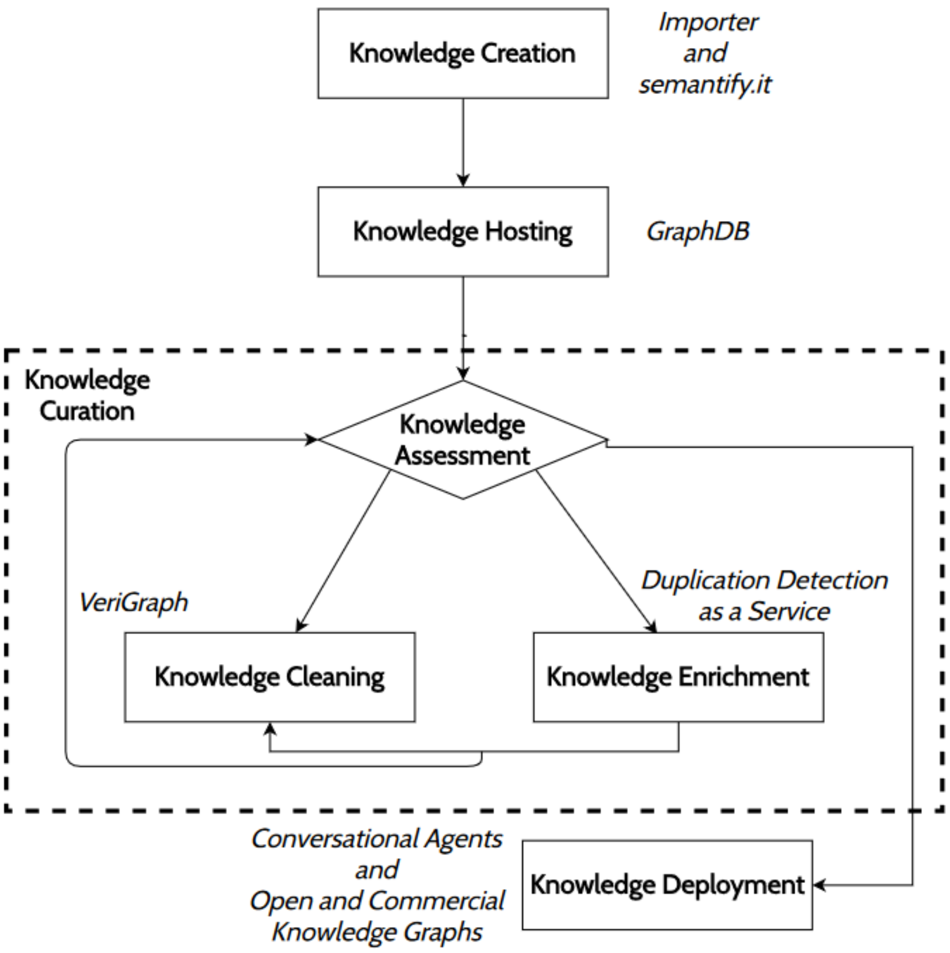
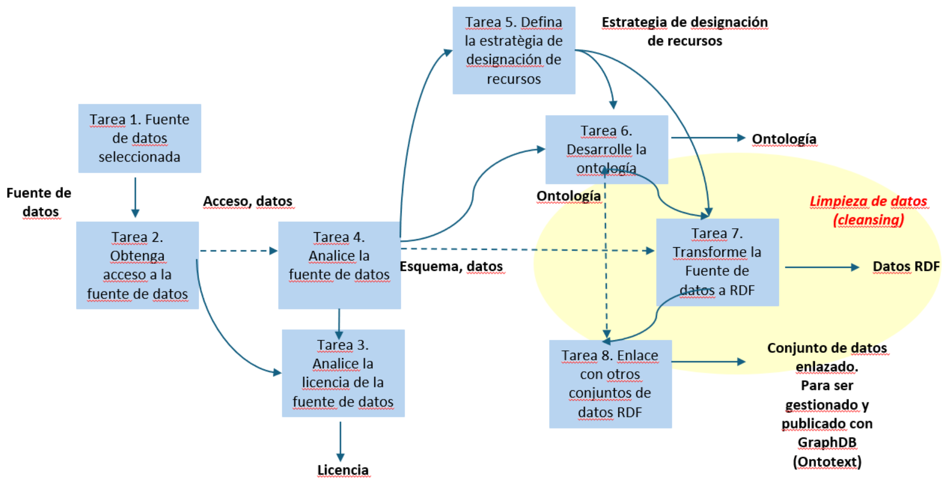


Ilustración 11: Simsek et al. (2021, p. 3)

- Radulovic et al. (2015)



Tarea 1. Fuente de datos seleccionada
Fuente de datos
Tarea 2. Obtenga acceso a la fuente de datos
Acceso, datos
Tarea 4. Analice la fuente de datos
Tarea 3. Analice la licencia de la fuente de datos
Licencia
Tarea 5. Defina la estratègia de designación de recursos
Estrategia de designación de recursos
Tarea 6. Desarrolle la ontología
Ontología
Ontología
Esquema, datos
Limpieza de datos (cleansing)
Tarea 7. Transforme la Fuente de datos a RDF
Datos RDF
Tarea 8. Enlace con otros conjuntos de datos RDF
Conjunto de datos enlazado. Para ser gestionado y publicado con GraphDB (Ontotext)

Ilustración 12: Radulovic et al., 2015, p. 180

Process 1: Knowledge Graph Implementation

The knowledge graph implementation process aims to construct and validate the knowledge graph. It is composed by a set of sub-processes that transform the input data, which can be of any type and format (e.g., tabular in CSV, text in PDF, etc.), into the knowledge graph and its later validation over a set of constraints.

Within the context of LOT4KG, and according to LOT for Knowlege Graphs Construction (2024), the output of this process is the implemented knowledge graph (virtual or materialized) and the associated rules for constructing (e.g., RML, SPARQL-Anything, etc.) and validating it (e.g., ShEx or SHACL shapes). The latter may also include a validation report as output.

Simsek et al al. (2021, pp. 3-5): "Knowledge Curation".

1.1. Data preparation

Objective: Application of data cleaning mechanisms to adapt them to the requirements of transforming the data source to RDF: completion, segmentation, deletion, etc.

Making initial transformations to the data to correct errors and adjust the columns and values of the data source to the requirements for transformation to RDF.

Resources:

- Data cleaning recipes: *LIMPIEZA DE DATOS CON ONTOTEXT REFINE*
- Tool: *Open Refine*

Output: Ready data source.

1.2. Mapping generation

Objective: According to Pernisch et al. (2025, pp. 287-288), during KG construction, we generate relationships between heterogeneous data sources and ontology terms using mapping languages (e.g., *RML* or *SPARQLAnything*).

Resources:

- Main resource: RDF mapping module within *Open Refine*. It provides three options.
  - Option 1: Using the Visual RDF Mapper.
  - Option 2: Using the SPARQL Query Editor.
  - Option 3: Using the Virtual SPARQL Endpoint. The user can specify the context using a SPARQL GRAPH clause and can even define it dynamically from values in the data. This should be the preferred one.
- Complementary resources:
  - *RML Ontology Modules* (W3C Community Group on Knowledge Graph Construction, 2023)
  - *R2RML: RDB to RDF Mapping Language* (W3C, 2012)
  - *SPARQLAnything* (2025)

Output: Mappings consisting of establishing correspondence between the records of the tables of relational databases and the classes and properties of the RDF dataset.

1.3. Data transformation

Objective: To convert data from the source databases to RDF through the ontology and the resource designation strategy. The converted data are inserted into one or more files in an RDF serialization format (especially, Turtle).

The setting of this subprocess depends on the format of the data (database, spreadsheets, etc.) and on the specific needs of the transformation process (e.g., dynamicity, scalability).

Resources:

There are tools available for transforming data from data streams, databases, XML files, spreadsheets… Our priority is to do it from data streams, as this will enable dynamic generation.

- Main resource: *Open Refine* (using the Virtual SPARQL Endpoint). Within Data Integration Using the Virtual SPARQL Endpoint the user can specify the context using a SPARQL GRAPH clause and can even define it dynamically from values in the data.
- Complementary resources:
  - From data streams: e.g., morph-RDB (2014/2024), D2R Server (2020).
  - From databases: e.g., morph-RDB (2014/2024), D2R Server (2020), TopBraid Composer (2025); OpenRefine (using the Visual RDF Mapper, SPARQL Query Editor o Virtual SPARQL Endpoint).
  - From XML files: e.g., XML2RDF (2018/2022), OpenRefine (using the Visual RDF Mapper, SPARQL Query Editor o Virtual SPARQL Endpoint).
  - From spreadsheets: e.g., Excel2rdf (2013/2013), RDF123 (2025), XLWrap (2009), TopBraid Composer (2025), OpenRefine (using the Visual RDF Mapper, SPARQL Query Editor o Virtual SPARQL Endpoint).

Output: One or more files in an RDF serialization format (especially, Turtle) to be inserted into a dataset repository.

1.4. Constraints generation

Objective: According to Pernisch et al. (2025, pp. 287-288), to generate SHACL shapes, which impose constraints on the shape of the KG.

According to Gayo et al (2018), ShEx and SHACL share the same goal, to have a mechanism for describing and validating RDF data using a high-level language, so there are a lot of common features that both share. Although both languages share a common goal, their designs are based on different approaches.

The designers of ShEx intended the language to be like a grammar or schema for RDF graphs. This design was inspired by languages such as Yacc, RelaxNG, and XML Schema. The main goal was to describe RDF graph structures so they could be validated against those descriptions.

In contrast, the designers of SHACL aimed at providing a constraint language for RDF. The main goal of SHACL is to verify that a given RDF graph satisfies a collection of constraints. In this sense, SHACL follows the Schematron approach, applied to RDF: it declares constraints that RDF graphs must fulfill. Just as Schematron relies strongly on XPath, SHACL relies strongly on SPARQL.

This difference is reflected in how validation results fit in. ShEx implementations usually construct a data structure representing the RDF graph that were validated, containing the nodes and shapes that were matched. After ShEx validation, the result shape map contains a structure which can be considered as an annotated graph that can be traversed or used for further actions, such as transforming RDF graphs into other data structures. This structure is analogous to the Post Schema Validation Infoset from XML Schema (see Section 3.1.3).

In contrast, SHACL describes in detail the errors returned when constraints are not satisfied. A SHACL validation report (see Section 5.5) can be very useful for detecting and repairing errors in RDF graphs. When there are no errors, SHACL processors usually report a single value, sh:conformance true. With SHACL, it can be difficult for users to distinguish the case in which a node is valid because it was checked against some shape, versus the case in which a node is not valid but was ignored by the SHACL processor because it was not reached during the validation process.

The SHACL recommendation prescribes a basic structure for each violation result but does not prescribe what information is to be returned when a node is validated. Nevertheless, SHACL processors can enrich their results. Shaclex, for example, returns information about the nodes validated.

- ShEx and SHACL can both be used to validate RDF.
- The expressiveness of ShEx and SHACL for common use cases is similar.
- ShEx is a W3C Community Group specification while SHACL Core and SHACL-SPARQL are a W3C Recommendation
- ShEx is schema-oriented, while SHACL is focused on defining constraints over RDF graphs.
- ShEx can be used with a compact syntax, a JSON-LD syntax, or any RDF syntax. SHACL can be used with any RDF syntax, and a draft compact syntax has been proposed.
- ShEx has support for recursion and cyclic data models while recursion in SHACL is undefined.
- SHACL has support for arbitrary SPARQL property paths while ShEx has support only for incoming and outgoing arcs.
- Both ShEx and SHACL support violation reporting at the shape level. For simple shapes, SHACL can further distinguish the violations per constraint, as well as provide more violation metadata. SHACL returns the violations in RDF using the Validation Report vocabulary while ShEx returns a shape map with all nodes that were validated, including the ones that pass validation while SHACL only the ones that failed.
- ShEx has a language agnostic extension mechanism called semantic actions while SHACL offers extensibility through SPARQL and JavaScript.

Resources:

- Specification: *Shapes Constraint Language (SHACL)* (W3C, 2017)

- Editing tools:
    - Main resource: SHACL Playground (2017): SHACL Playground is a web-based tool for experimenting with SHACL rules. It allows you to create and edit SHACL shapes, validate RDF data against them, and view the validation report.
    - Complementary resources:
        - Apache Jena (2025): Apache Jena is a Java-based framework for building Semantic Web applications. It includes a SHACL implementation that allows you to validate RDF data against SHACL rules.
        - OntoPad (2020/2024)
        - Protégé shacl-plugin (2015/2025)
        - shaclEditor (2020/2024)
        - TopBraid Composer (2025): TopBraid Composer is a graphical tool for working with RDF data and ontologies. It includes a SHACL editor that allows you to create and edit SHACL shapes visually.

Output: Constraints.

1.5. Data validation

Objective: Validation with the W3C standard Shapes Constraint Language (SHACL) standard is a valuable tool for efficient data consistency checking. It is useful in efforts towards data integration, as well as examining data compliance — for example, every GeoName URI must start with https://sws.geonames.org/\d+/, or that age values must be above 18 years.

The language validates RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. In SHACL, RDF graphs that are used in this manner are called shapes graphs, and the RDF graphs that are validated against a shapes graph are called data graphs.

Resource: *Ontotext GraphDB SHACL validation*. It is supported by GraphDB via RDF4J's ShaclSail storage and interface layer. https://graphdb.ontotext.com/documentation/10.8/shacl-validation.html

Output: Validation report and a refined version of the KG.

Process 2: Knowledge Graph Enrichment

Objectives: Enhancing the knowledge graph with data from external data. It includes two subprocesses: knowledge extraction and knowledge fusion.

2.1. Knowledge extraction

Reference methodologies: Dong et al. (2014), Hoyt et al. (2019), Blumauer & Nagy (Blumauer & Nagy, 2020) and Hogan et al. (2022).

It consists in TRANSFORMING SEMISTRUCTURED AND UNSTRUCTURED DATA INTO RDF to amplify the entities and ontology within our knowledge graph. There are three source types.

(1) Text documents (TXT or similar)
(2) Markup sources
- Wrapper-based extraction

- DOM trees (DOM)
- Web tables (TBL)/ Web table extraction
- Web annotations (ANO)
- Deep web crawling

(3) Structured Sources (Much of the legacy data available within organisations and on the Web is represented in structured formats, primarily tables – in the form of relational databases, CSV files, etc. – but also tree-structured formats such as JSON, XML etc.)

- Mapping from tables
- Mapping from trees
- Mapping from other knowledge graphs

For the extraction, there are two different types depending on the source types.

For the type 1, the extraction process includes:

- Model 1:
  - Pre-processing
  - Named Entity Recognition (NER)
  - Entity Linking (EL)
  - Relation Extraction (RE)
  - Joint tasks
- Model 2:
  - Extracting Semantic Frames from Specialized Corpora using corpus tools: MWE toolkit and Sketch Engine (Sánchez Cárdenas, 2024).
- Model 3:
  - Extracting Semantic Frames from Large language models (LLMs): (Ananya et al., 2024); (Cauter & Yakovets, 2024); (Chepurova et al., 2024); (Papaluca et al., 2024); (Zhang et al., 2024).

For the types 2 and 3, the extraction process includes:

- triple identification;
- entity linkage;
- predicate linkage

2.2. Data fusion

Reference methodologies: Bleiholder et al (2009) and Dong et al. (2014).

It is the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation. Knowledge graph fusion is the task of constructing a unified knowledge graph from different data sources. The ontology that defines the schema of the data helps here as well, when mapping structured data and also when extracting facts from unstructured data. Together with machine learning approaches, this will even enable the establishment of automated mapping of structured information and quality checks on the data itself. In addition, it will enable us to recognize more facts and information in unstructured data and make them more valuable.

Return to subprocesses of 1.6. Constraints generation and 1.7. Data validation within Process 1.

Process 3: Linking with Other Datasets

The task of data linking has the goal of creating links in the RDF data and it can be achieved in several consecutive steps by using the RDF data set and the ontology as inputs.

Subprocesses, according to Radulovic (2015)**:**

3.1. To identify classes whose instances can be the subject of linking.
3.2. To identify data sets that may contain instances for the previously identified classes.

3.3. To select the tools for performing the task. Different tools for data linking exist, and each tool has its advantages and provides different functionalities for certain matching tasks. However, in some cases, the linking can be performed manually (e.g., when the generated data set is small, or when the number of instances to link is low), and the next step is not necessarily performed.

Main resource: reconciliation service of the RDF extension within OpenRefine.

3.4. To use the tool in order to obtain links. Linking between resources within our knowledge graph and the resources within the external dataset is expressed by the property *owl:sameAs*.

Process 4: Knowledge Graph Publication

Objectives: The high-level KG publication process is a counterpart to the ontology publication process. It captures the necessary tasks and steps which are taken to document and make the KG available online. The knowledge graph publication will follow the model of Collection of Data to promote its responsible computational use. To achieve this, the checklist for publishing Collections as data in GLAM institutions, according to SSH Open Marketplace (2022) and Candela et al. (2023), will be applied. This checklist includes 11 items:

1. Provide a clear license allowing reuse of the dataset without restrictions (e.g., CC0, CC BY)
2. Provide a suggestion of how to cite your dataset
3. Include documentation about the dataset
4. Use a public platform to publish the dataset
5. Share examples of use as additional documentation
6. Give structure to the dataset
7. Provide machine-readable metadata (about the dataset itself)
8. Include your dataset in collaborative edition platforms
9. Offer an API to access your repository.
10. Develop a portal page
11. Add a terms of use

It is important to note the following:

- For items 1, 4, 8, 9, and 10, the agreement of the original database owners (scenarios) is required.
- For item 9, in addition to the API, a SPARQL endpoint must be provided.

The lower-level processes part of the KG publication are documentation and data publication. During the documentation step, the mappings, RDF data, SHACL shapes and validation report are used to document the process and output of the implementation. The output is the HTML documentation, which can then be published during the data publication step alongside the online KG.

Resources:

- Main resources:
- To store and publish Linked Open Data
    - *Wilibase* (Wikimedia Deutschland, 2025)
- Checklist for publishing Collections as data in GLAM institute
    - Candela et al. (2023)
    - SSH Open Marketplace (2022)
- To document the KG
    - *Data Catalog Vocabulary (DCAT)* (W3C, 2024)
    - *RML mapping documentation* (2023/2024)
- To make the KG available online
    - Ali et al. (2022)
    - *OBA: An Ontology-Based Framework for Creating REST APIs for Knowledge Graphs* (2019/2021)

  Output: The documented KG and the online accessible resource.

Process 5: Knowledge Deployment

Simsek et al al. (2021, pp. 3-5): "Knowledge Deployment".

The knowledge deployment task deals with the applications that are powered by a knowledge graph.

Output: This process involves utilizing applications that are powered by a knowledge graph. The outcome includes a knowledge graph available in a repository with a SPARQL endpoint for data consumption by external datasets. Additionally, it will be accessible via:

- Data dumps (RDF-XML/Turtle/JSON-LD, etc.).

- REST API.

- A web platform, implemented through Ontotext GraphDB, Wikibase, or a custom-developed solution.

Ontotext GraphDB Process 6: Knowledge Graph Maintenance

This task is modelled along the same lines as the ontology maintenance task. It is specifically aimed at fixing bugs in the already published KG. This step does not capture proper evolution, the changes within the ontology or the data sources. Therefore, the detailed process is named bug detection. The output of the process is the issues and bugs to be fixed by backtracking in the process.

Additionally, it is crucial to consider backward compatibility when making changes to an ontology. Modifications should not disrupt the existing data structure or any APIs that depend on the ontology. Failure to ensure compatibility may lead to broken integrations, data inconsistencies, or the need for extensive refactoring in downstream applications. A well-defined strategy for versioning and deprecation of obsolete elements should be in place to minimize disruptions. In this sense, it is essential to analyze the impact of changes in the class taxonomy, modifications of properties, and adjustments to SHACL constraints, as these elements directly influence data validation, reasoning processes, and external system interactions.

4.1. Change Detection and Impact Analysis Step

To be able to update an already existing KG, the changes applied to the ontology need to be examined and analysed against the KG.

We define three sub-processes:

- detect delta, whose output are formalised changes.
- refine relevant changes, whose output are relevant changes.
- analyse change impact, whose output is the impact report.

Only the middle process is mandatory.

Resource:

- Hartung et al. (2013)

Output: The main output of this process is a list of relevant changes, relevant for the update of the KG.

4.2. Knowledge Graph Update

This process, like the ontology update process, reflects the implementation process of the knowledge graph (KG). Its purpose is to ensure the KG is current with any changes in the ontology or source data. Hence, the KG update process can be triggered:

- From the change detection processes which provide the list of ontology changes. See Process 1: Knowledge Graph Implementation and Process 2: Knowledge Graph Enrichment.
- From changes to the source data, depicted by the arrow connecting the KG maintenance process with KG update. See Process 3: Ontology Implementation.

Version control requires the development of three types of activities:

- Regular generation of versions of the ontology and knowledge graph in one of the formats mentioned in Process 5.
- Formal description of the different versions of the ontology and knowledge graph using Prov-O (W3C, 2013).
  Implementation of tools or mechanisms to visualize and monitor changes between versions of an ontology and the knowledge graph. Clear examples include OntoDiffGraph (Lara et al., 2017) y KGdiff (CMACH508, 2023/2025), respectively.

The KG update process has four sub-processes that make it possible to update the KG, each of them associated with the corresponding assets of the KG implementation:

- Mapping update, whose output are mappings.
- Data update, whose output is RDF data graph.

- Constraints update, whose output are constraints.
- Data validation, whose output is a validation report.

Resources:

- Conde-Herreros et al. (2024)
- Van Assche et al. (2024)

Output: The high-level output is the updated RDF graph and its associated assets (i.e. mappings, data constraints, and validation report), which are to be published using the KG publishing process.

Bibliography

agnieszkalawrynowicz. (2017). *Agnieszkalawrynowicz/tddonto* [Java]. https://github.com/agnieszkalawrynowicz/tddonto (Obra original publicada en 2016)

AKSW/OntoPad. (2024). [Vue]. AKSW Research Group @ University of Leipzig. https://github.com/AKSW/OntoPad (Obra original publicada en 2020)

Ali, Waqas, Saleem, Muhammad, Yao, Bin, Hogan, Aidan, & Ngomo, Axel-Cyrille Ngonga. (2022). A survey of RDF stores & SPARQL engines for querying knowledge graphs. *The VLDB Journal*, *31*(3), 1-26. https://doi.org/10.1007/s00778-021-00711-3

Ananya, Ananya, Tiwari, Sanju, Mihindukulasooriya, Nandana, Soru, Tommaso, Xu, Ziwei, & Moussallem, Diego. (2024). Towards Harnessing Large Language Models as Autonomous Agents for Semantic Triple Extraction from Unstructured Text. *Extended Semantic Web Conference*. https://ceur-ws.org/Vol-3747/text2kg_paper1.pdf

Apache Jena. (2025). https://jena.apache.org/

Archivo: Ontology Archive. (2025). Dbpedia. https://archivo.dbpedia.org/list#list

Atlassian. (2025a). *Bitbucket: Git solution for teams using Jira*. Bitbucket. https://bitbucket.org/product

Atlassian. (2025b). *Jira: Issue & Project Tracking Software*. https://www.atlassian.com/software/jira?campaign=18455741485&adgroup=137008204930&targetid=kwd-855725830&matchtype=e&network=g&device=c&device_model=&creative=68797292

7818&keyword=jira&placement=&target=&ds_eid=700000001558501&ds_e1=GOOGLE&gad_source=1&gclid=Cj0KCQiAwtu9BhC8ARIsAI9JHaIsIuVb9-tA5fA4gAsX3uaOtrwvdzbJaz-sqFILFD40BgMRv4mdMhcaAkNDEALw_wcB

BARTOC.org. (2020). https://bartoc.org/

Bleiholder, Jens, & Naumann, Felix. (2009). Data fusion. *ACM Computing Surveys*, *41*(1), 1-41. https://doi.org/10.1145/1456650.1456651

Blumauer, Andreas, & Nagy, Helmut. (2020). *The Knowledge Graph Cookbook*. mono/monochrom.

Candela, Gustavo, Gabriëls, Nele, Chambers, Sally, Dobreva, Milena, Ames, Sarah, Ferriter, Meghan, Fitzgerald, Neil, Harbo, Victor, Hofmann, Katrine, & Holownia, Olga. (2023). *A Checklist to Publish Collections as Data in GLAM Institutions*. https://core.ac.uk/download/pdf/611876137.pdf

Cauter, Zeno, & Yakovets, Nikolay. (2024). Ontology-guided Knowledge Graph Construction from Maintenance Short Texts. *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, 75-84. https://doi.org/10.18653/v1/2024.kallm-1.8

Centelles, Miquel, & Ferran-Ferrer, Núria. (2024). Assessing knowledge organization systems from a gender perspective: Wikipedia taxonomy and Wikidata ontologies. *Journal of Documentation*, *80*(7), 124-147.

Centelles, Miquel, & Ferrer, Núria Ferran. (2024). Taxonomies and ontologies in Wikipedia and Wikidata: An in-depth examination of knowledge organization systems. *Hipertext. net*, *28*, 33-48.

Chepurova, Alla, Kuratov, Yurii, Bulatov, Aydar, & Burtsev, Mikhail. (2024). Prompt Me One More Time: A Two-Step Knowledge Extraction Pipeline with Ontology-Based Verification. *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, 61-77. https://aclanthology.org/2024.textgraphs-1.5/

CMACH508. (2025). *CMACH508/KGDiff* [Jupyter Notebook]. https://github.com/CMACH508/KGDiff (Obra original publicada en 2023)

CommonCoreOntology/CommonCoreOntologies. (2025). [Makefile]. CommonCoreOntology. https://github.com/CommonCoreOntology/CommonCoreOntologies (Obra original publicada en 2017)

Conde-Herreros, Diego, Stork, Lise, Pernisch, Romana, Poveda-Villalón, María, Corcho, Oscar, & Chaves-Fraga, David. (2024). *Propagating Ontology Changes to Declarative Mappings in Construction of Knowledge Graphs*. KGCW'24: 5th International Workshop on Knowledge Graph Construction, May 26th, 2024, Crete, Greece.

D2R Server. (2020). [The D2RQ Platform]. http://d2rq.org/d2r-server

Dong, Xin Luna, Gabrilovich, Evgeniy, Heitz, Geremy, Horn, Wilko, Murphy, Kevin, Sun, Shaohua, & Zhang, Wei. (2014). From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, *7*(10), 881-892. https://doi.org/10.14778/2732951.2732962

Ekaputra, Fajar J. (2025). *Fekaputra/shacl-plugin* [Java]. https://github.com/fekaputra/shacl-plugin (Obra original publicada en 2015)

Fernández Izquierdo, Alba. (2020). *THEMIS: Validate your ontology*. https://themis.linkeddata.es/

Fernández-Izquierdo, Alba, Poveda-Villalón, María, & García-Castro, Raúl. (2017). *CORAL: A corpus of ontological requirements annotated with Lexico-Syntactic Patterns*. https://coralcorpus.linkeddata.es/

FluentEditor: Ontology Editor. (2025). Cognitum. https://www.cognitum.eu/semantics/fluenteditor/

Garijo, Daniel. (2017). *WIDOCO: A wizard for documenting ontologies* [JavaScript]. https://doi.org/10.1007/978-3-319-68204-4_9 (Obra original publicada en 2013)

Gayo, Jose Emilio Labra, Prud'hommeaux, Eric, Boneva, Iovka, & Kontokostas, Dimitris. (2018). *Validating RDF Data*. Springer International Publishing. https://doi.org/10.1007/978-3-031-79478-0

GFZ German Research Centre For Geosciences, Humboldt-Universität Zu Berlin, Germany Karlsruhe Institute Of Technology (KIT), Purdue University Libraries, Bertelmann, Roland, Buys, Matt, Cousijn, Helena, Dierolf, Uwe, Elger, Kirsten, Fenner, Martin, Ferguson, Lea Maria, Fritze, Florian, Fuchs, Claudio, Goebelbecker, Hans-Jürgen, Gundlach, Jens, Kindling, Maxi, Kloska, Gabriele, Klump, Jens, Kramer, Claudia, … van de Sandt, Stephanie. (2013). *Registry of Research Data Repositories*. DataCite. https://doi.org/10.17616/R3D

GitHub: Build and ship software on a single, collaborative platform. (2025). GitHub. https://github.com/

GitLab: The most-comprehensive AI-powered DevSecOps platform. (2025). https://about.gitlab.com/

Graf.fo. (2015). Graf.Fo. https://gra.fo/

Hartung, Michael, Groß, Anika, & Rahm, Erhard. (2013). COnto–Diff: Generation of complex evolution mappings for life science ontologies. *Journal of Biomedical Informatics*, *46*(1), 15-32. https://doi.org/10.1016/j.jbi.2012.04.009

HermiT Reasoner: Home. (2009). http://www.hermit-reasoner.com/

Hogan, Aidan, Blomqvist, Eva, Cochez, Michael, d'Amato, Claudia, Melo, Gerard de, Gutierrez, Claudio, Gayo, José Emilio Labra, Kirrane, Sabrina, Neumaier, Sebastian, Polleres, Axel, Navigli, Roberto, Ngomo, Axel-Cyrille Ngonga, Rashid, Sabbir M., Rula, Anisa, Schmelzeisen, Lukas, Sequeda, Juan, Staab, Steffen, & Zimmermann, Antoine. (2022). Knowledge Graphs. *ACM Computing Surveys*, *54*(4), 1-37. https://doi.org/10.1145/3447772

Hoyt, Charles Tapley, Domingo-Fernández, Daniel, Aldisi, Rana, Xu, Lingling, Kolpeja, Kristian, Spalek, Sandra, Wollert, Esther, Bachman, John, Gyori, Benjamin M., Greene, Patrick, & Hofmann-Apitius, Martin. (2019). *Re-curation and rational enrichment of knowledge graphs in Biological Expression Language*. *2019*.

Lara, André, Henriques, Pedro Rangel, & Gançarski, Alda. (2017). *OntoDiffGraph*. https://epl.di.uminho.pt/~gepl/GEPL_DS/OntoDiffGraph/

Linked Open Vocabularies (LOV). (2025). https://lov.linkeddata.es/dataset/lov/

LOT4KG: LOT for Knowlege Graphs Construction. (2024). https://lot.linkeddata.es/LOT4KG/

MaastrichtU-IDS/xml2rdf. (2022). [Java]. Maastricht University IDS. https://github.com/MaastrichtU-IDS/xml2rdf (Obra original publicada en 2018)

Mantis Bug Tracker. (2025). https://mantisbt.org/

Noy, Natalya F., & McGuinness, Deborah L. (2001). *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and …. https://corais.org/sites/default/files/ontology_development_101_aguide_to_creating_your_first_ontology.pdf

Oeg-upm/LOT-resources. (2025). [HTML]. Ontology Engineering Group (UPM). https://github.com/oeg-upm/LOT-resources (Obra original publicada en 2020)

Oeg-upm/morph-rdb. (2024). [Scala]. Ontology Engineering Group (UPM). https://github.com/oeg-upm/morph-rdb (Obra original publicada en 2014)

Ontohub. (2025). https://ontohub.org/

OnToology. (2025). https://ontoology.linkeddata.es/

Ontotext GraphDB. (2025). Ontotext. https://www.ontotext.com/products/graphdb/

OOPS!: OntOlogy Pitfall Scanner! (2023). https://oops.linkeddata.es/

Osorio, Maximiliano, & Garijo, Daniel. (2021). *OBA: An Ontology-Based Framework for Creating REST APIs for Knowledge Graphs* [HTML]. https://doi.org/10.1007/978-3-030-62466-8_4 (Obra original publicada en 2019)

Papaluca, Andrea, Krefl, Daniel, Rodríguez Méndez, Sergio, Lensky, Artem, & Suominen, Hanna. (2024). Zero- and Few-Shots Knowledge Graph Triplet Extraction with Large Language Models. *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, 12-23. https://doi.org/10.18653/v1/2024.kallm-1.2

Pernisch, Romana, Poveda-Villalón, María, Conde-Herreros, Diego, Chaves-Fraga, David, & Stork, Lise. (2025). When Ontologies Met Knowledge Graphs: Tale of a Methodology. En Albert Meroño Peñuela, Oscar Corcho, Paul Groth, Elena Simperl, Valentina Tamma, Andrea Giovanni Nuzzolese, Maria Poveda-Villalón, Marta Sabou, Valentina Presutti, Irene Celino, Artem Revenko, Joe Raad, Bruno Sartini, & Pasquale Lisena (Eds.), *The Semantic Web: ESWC 2024 Satellite Events* (Vol. 15344, pp. 286-290). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-78952-6_43

Peroni, Silvio. (2011). *Graffoo: Graphical Framework for OWL Ontologies*. https://essepuntato.it/graffoo/

Peroni, Silvio. (2013). *LODE: Live OWL Documentation Environment*. https://essepuntato.it/lode/

PoolParty Semantic Suite. (2025). rty. https://www.poolparty.biz/

Poveda-Villalón, María. (2024). *Chowlk*. The Semantic Web. https://chowlk.linkeddata.es/

Poveda-Villalón, María, Fernández-Izquierdo, Alba, Fernández-López, Mariano, & García-Castro, Raúl. (2022a). LOT: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, *111*, 104755. https://doi.org/10.1016/j.engappai.2022.104755

Poveda-Villalón, María, Fernández-Izquierdo, Alba, Fernández-López, Mariano, & García-Castro, Raúl. (2022b). LOT: An industrial oriented ontology engineering framework.

Engineering Applications of Artificial Intelligence, *111*, 104755. https://doi.org/10.1016/j.engappai.2022.104755

Protégé. (2025). https://protege.stanford.edu/

Pylode. (2024). MyNixOS. https://mynixos.com/nixpkgs/package/pylode

Radulovic, Filip, Poveda-Villalón, María, Vila-Suero, Daniel, Rodríguez-Doncel, Víctor, García-Castro, Raúl, & Gómez-Pérez, Asunción. (2015). Guidelines for Linked Data generation and publication: An example in building energy consumption. *Automation in Construction*, *57*, 178-187. https://doi.org/10.1016/j.autcon.2015.04.002

RDF123. (2025). UMBC ebiquity. https://ebiquity.umbc.edu/project/html/id/82/RDF123

Sánchez Cárdenas, Beatriz. (2024). Extracting Semantic Frames from Specialized Corpora for Lexicographic Purposes. *Círculo de Lingüística Aplicada a La Comunicación*, *99*, 163-177. https://doi.org/10.5209/clac.90626

Sarker, Md Kamruzzaman. (2023). *OWLAx* [Java]. https://github.com/md-k-sarker/OWLAx (Obra original publicada en 2016)

SHACL Playground. (2017). https://shacl.org/playground/

Simsek, Umutcan, Angele, Kevin, Kärle, Elias, Opdenplatz, Juliette, Sommer, Dennis, Umbrich, Jürgen, & Fensel, Dieter. (2021). Knowledge Graph Lifecycle: Building and Maintaining Knowledge Graphs. *KGCW@ ESWC*. https://ceur-ws.org/Vol-2873/paper12.pdf

SPARQL anything. (2025). https://sparql-anything.cc/

SSH Open Marketplace. (2022). *A workflow to publish Collections as Data: The case of Cultural Heritage data spaces*. Social Sciences & Humanities Open Marketplace. https://marketplace.sshopencloud.eu/workflow/I3JvP6

Stardog-union/pellet. (2025). [Web Ontology Language]. Stardog Union. https://github.com/stardog-union/pellet (Obra original publicada en 2013)

The OntoCheck Plugin. (2011). https://www2.imbi.uni-freiburg.de/ontology/OntoCheck/

Toledo, Jhon, Chaves, David, Iglesias-Molina, Ana, & Garijo, Daniel. (2024). *RML mapping documentation* [Python]. https://github.com/oeg-upm/rmldoc (Obra original publicada en 2023)

Topbraid Composer. (2025). AllegroGraph. https://allegrograph.com/topbraid-composer/

Valdestilhas, Andre. (2024). *Firmao/shaclEditor* [JavaScript]. https://github.com/firmao/shaclEditor (Obra original publicada en 2020)

Van Assche, Dylan, Rojas, J., De Meester, Ben, & Colpaert, Pieter. (2024). *IncRML: Incremental knowledge graph construction from heterogeneous data sources*. https://www.semantic-web-journal.net/system/files/swj3674.pdf

VocBench: A Collaborative Management System for OWL ontologies, SKOS(/XL) thesauri, Ontolex-lemon lexicons and generic RDF datasets. (2017). https://vocbench.uniroma2.it/

Vocol/vocol. (2023). [JavaScript]. VoCol - Vocabulary collaboration and build environment. https://github.com/vocol/vocol (Obra original publicada en 2014)

W3C. (2008). *Best Practice Recipes for Publishing RDF Vocabularies*. https://www.w3.org/TR/swbp-vocab-pub/

W3C. (2012). *R2RML: RDB to RDF Mapping Language*. https://www.w3.org/TR/r2rml/

W3C. (2013). *PROV-O: The PROV Ontology*. https://www.w3.org/TR/2013/REC-prov-o-20130430/

W3C. (2017, julio 20). *Shapes Constraint Language (SHACL)*. W3C. https://www.w3.org/TR/shacl/

W3C. (2024). *Data Catalog Vocabulary (DCAT)*. https://www.w3.org/TR/vocab-dcat-3/

W3C. (2025). *Semantic Web Standards*. Making the Web work. https://www.w3.org/2001/sw/wiki/Main_Page

W3C Community Group on Knowledge Graph Construction. (2023). *RML Ontology Modules*. https://kg-construct.github.io/rml-resources/portal/

waqarini. (2013). *Waqarini/excel2rdf* [Java]. https://github.com/waqarini/excel2rdf (Obra original publicada en 2013)

WebVOWL. (2025). https://service.tib.eu/webvowl/

Wikimedia Deutschland. (2025). *Wikibase*. Wikibase. https://wikiba.se

XLWrap: Spreadsheet-to-RDF Wrapper. (2009). https://xlwrap.sourceforge.io/

Zhang, Yujia, Sadler, Tyler, Taesiri, Mohammad Reza, Xu, Wenjie, & Reformat, Marek. (2024). Fine-tuning Language Models for Triple Extraction with Data Augmentation. *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, 116-124. https://aclanthology.org/2024.kallm-1.12/