



Report D4.2

"Methodological framework for ontology management"

Grant Agreement: 958371



OntoCommons - Ontology-driven data documentation for Industry Commons, has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 958371.

Project Title	Ontology-driven data documentation for Industry Commons
Project Acronym	OntoCommons
Project Number	958371
Type of project	CSA - Coordination and support action
Topics	DT-NMBP-39-2020 - Towards Standardised Documentation of Data through taxonomies and ontologies (CSA)
Starting date of Project	01 November 2020
Duration of the project	36 months
Website	www.ontocommons.eu

Report D4.2

"Methodological framework for ontology management"

Work Package	Ontology Commons Ecosystem Toolkit
Task	Methodological guidelines
Lead author	Alba Fernández-Izquierdo (UPM)
Contributors	María Poveda-Villalón (UPM), Emna Amdouni (ENIT)
Peer reviewers	Hedi Karray (ENIT)
Version	final
Date	23/09/2021

Keywords

Ontology; Ontology methodology

Disclaimer

OntoCommons.eu has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement no. 958371. The content of this document does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of such content. The European Commission is not liable for any use that may be made of the information contained herein.

Copyright notice © 2020 OntoCommons.eu Consortium.

Executive Summary

This document describes the Linked Open Terms (LOT) methodology, the methodological framework for OntoCommons. This document introduces the activities that should be performed in the ontology development process and includes recommendations and guidelines to put them into practice.

The document first presents the state of the art in ontology development methodologies, including well-known and agile approaches, which are considered as the basis for the LOT methodology. Subsequently, the document introduces the additional inputs considered for the definition of LOT for OntoCommons, involving the domain-specific semantic landscape and the workshop on domain ontologies to identify current challenges and gaps in ontology development, and the demonstrators' requirements on ontology tools and ontologies to identify existing requirements. Taking such inputs into account, an overview of LOT is presented.

The LOT methodology focuses on the reuse of terms existing in published ontologies and on the publication of the ontology according to the Linked Data principles. It is an iterative methodology that includes four main activities: ontology requirement specification, ontology implementation, ontology publication, and ontology maintenance.

Table of Contents

1. Introduction.....	6
2. Brief state of the art of Ontology Engineering methodologies.....	8
3. Input analysis	15
3.1 Domain ontologies development.....	15
3.2 Ontology landscape analysis.....	16
3.3 First focused workshop on domain ontologies.....	16
3.4 Demonstrators	16
4. LOT Methodology.....	18
4.1 Ontology requirements specification activity	19
4.1.1 Use case specification	20
4.1.2 Data exchange identification.....	21
4.1.3 Purpose and scope identification	21
4.1.4 Functional ontology requirement proposal.....	21
4.1.5 Functional ontology requirement completion.....	26
4.1.6 ORSD formalisation.....	26
4.1.7 Functional ontology requirements formalisation	28
4.2 Ontology implementation.....	28
4.2.1 Ontology conceptualisation.....	29
4.2.2 Ontology encoding	30
4.2.3 Ontology reuse	31
4.2.4 Ontology evaluation	32
4.3 Ontology publication	32
4.3.1 Documentation generation.....	33
4.3.2 Online publication	34
4.4 Ontology maintenance.....	34
4.4.1 Bug detection.....	35
4.4.2 New requirements proposal.....	35
5. Conclusions and future work	36
6. References.....	37

List of Figures

Figure 1 Overview of the relation between D4.2 with WP3, WP5, and other tasks in WP4	7
Figure 2 Procedure for ontology design and evaluation	8
Figure 3 METHONTOLOGY development process and life cycle	9
Figure 4 On-To-Knowledge development process and life cycle	10
Figure 5 DILIGENT life cycle.....	11
Figure 6 NeOn scenarios for building ontology networks.....	11
Figure 7 The building blocks of RapidOWL: Values, Principles, and Practices	12
Figure 8 AMOD Framework.....	13
Figure 9 Brief summary of SAMOD.....	13
Figure 10 Overview of LOT activities.....	19
Figure 11 Ontology requirements specification sub-activities.....	20
Figure 12 'Concepts' for ontology requirements	22
Figure 13 Table of 'Relations' for ontology requirements	23
Figure 14 'Concepts' to extract ontology requirements from MODA specifications.....	24
Figure 15 Table of 'Relations' to extract ontology requirements from MODA specifications	24
Figure 16 Table of 'Attributes' to extract ontology requirements from MODA specifications.....	24
Figure 17 Excerpt of ontology requirements	25
Figure 18 Template of the ORSD.....	27
Figure 19 Example of tests using the Verification Test Case ontology	28
Figure 20 Ontology implementation subactivities.....	29
Figure 21 Example of conceptualisation using the Chowlk notation	30
Figure 22 Ontology publication subactivities	33
Figure 23 Ontology maintenance sub-activities.....	35
Figure 24 Components of the ontology ecosystem toolkit.....	36

List of Tables

Table 1 Comparison of the state-of-the-art methodologies: "yes" indicates that the methodology totally covers the criterion, and "no" indicates that the methodology does not totally cover the criterion. We mention "n/a" when the methodology does not provide information about this criterion.....	15
Table 2 requirements- Application of ontologies.....	17
Table 3 requirements- Development of ontologies.....	17
Table 4 Information collected in the ontology long survey	40

1.Introduction

OntoCommons aims to develop a recommendation on the principles, best practices and methods for the development and maintenance of ontology. In that context, this report describes the Linked Open Terms (LOT) methodology, the methodological framework for the development and documentation of ontologies, including not only the activities that should be performed in the ontology development process, but also the guidelines, resources, and recommendations to support it.

LOT is an overall and lightweight methodology to build ontologies based on existing methodologies such as NeOn [1] and oriented to developments and technologies in the semantic web. The LOT methodology focuses on alignment with industrial development, in addition to academic and research projects and software development. It was first proposed by María Poveda-Villalón [2] and further developed in the European VICINTIY project [3]. Subsequently, it was also applied to European projects such as BIMERR [4] and DELTA [5]. This report describes the LOT methodology adapted and applied to the OntoCommons needs and particularities.

Figure 1 shows an overview of the inputs of this report, which includes tasks to consider current gaps and challenges in ontology development (report on 'Domain-specific semantic landscape' and 'Report on the first focused workshop on domain ontologies') and the needs of the use cases that should be supported (report on 'Requirements on ontology tools and ontologies'). Furthermore, as also shown in Figure 1, this report will serve as input for the implementation of the ontology ecosystem reference and for the ontology registry, as well as for the ontology knowledge graph and validation.

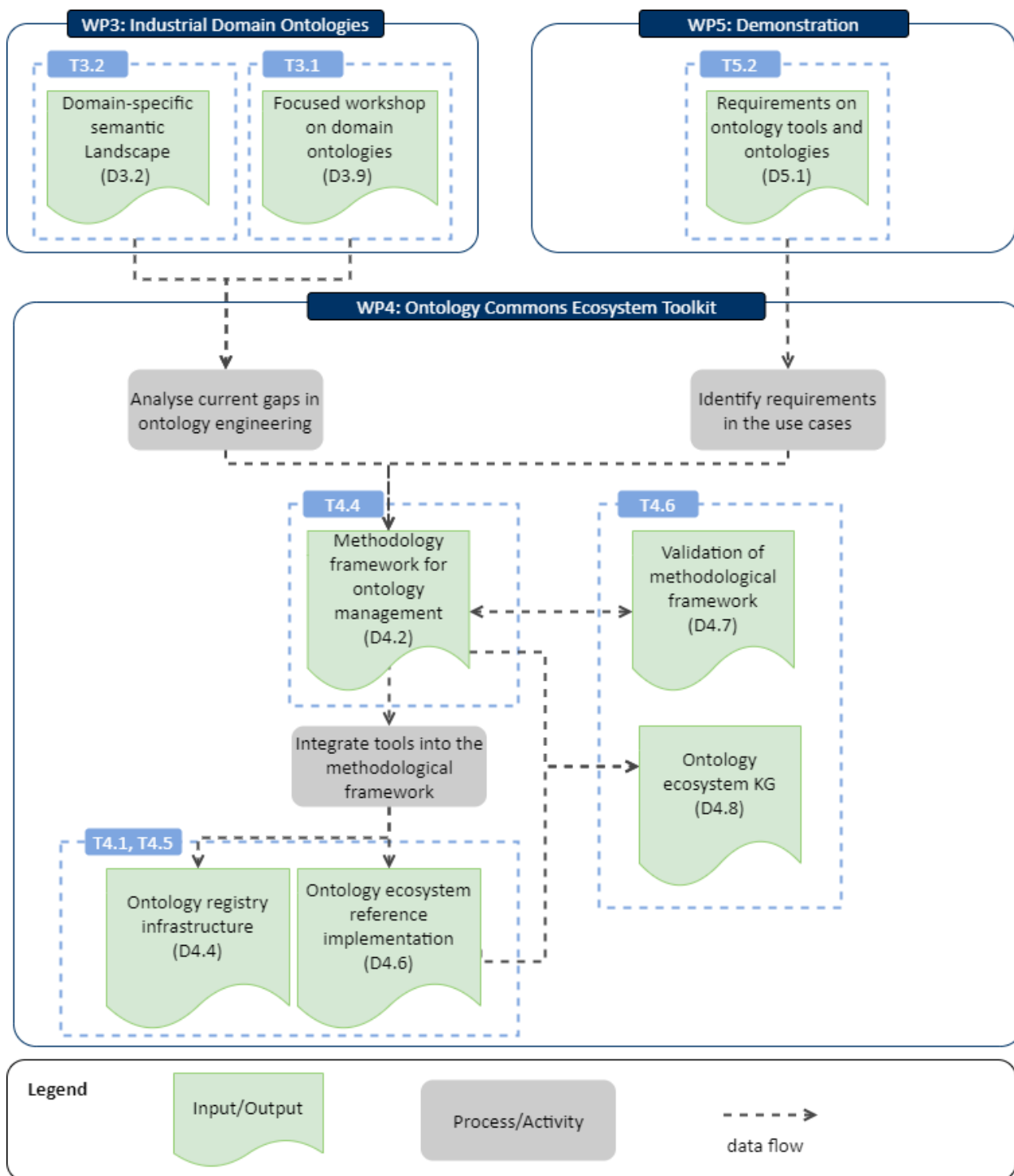


Figure 1 Overview of the relation between D4.2 with WP3, WP5, and other tasks in WP4

2. Brief state of the art of Ontology Engineering methodologies

The Ontology Engineering discipline investigates the principles, methods, and tools for initiating, developing, and maintaining ontologies. It provides life cycles that go from requirement definition to ontology maintenance, as well as methodologies, techniques, and tools to support and drive the development of ontologies.

This section includes a brief description of the main ontology development methodologies, namely, Grüninger & Fox [6], METHONTOLOGY [7], On-To-Knowledge [8], DILIGENT [9], NeOn [1], RapidOWL [10], SAMOD [11] and AMOD [12]. These methodologies are taken as the basis, and the core components are reused in the LOT methodology proposed in this report. The LOT methodology will be followed and extended when necessary for the OntoCommons use cases ontology developments.

The first methodology is the one proposed by Grüninger and Fox, which is based on the idea of common-sense models that can deduce answers to queries that require relatively shallow knowledge of the domain. The procedure for engineering such models is depicted in Figure 2. It should also be mentioned that this work introduced the notion of competency questions, which are widely used to extract ontology requirements and to test ontologies.

As in many ontology development methodologies, the competency question technique is included in the LOT methodology.

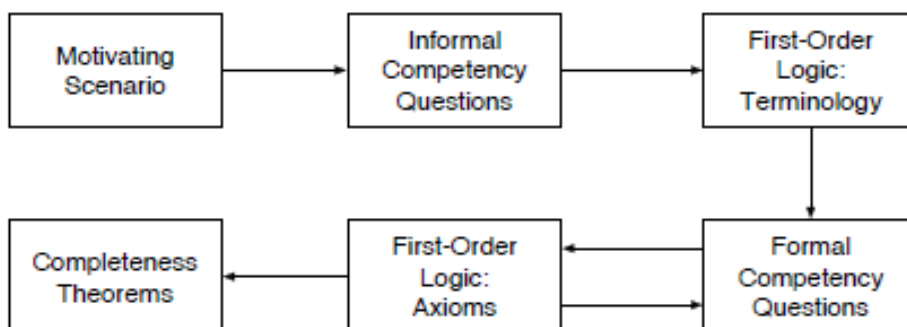


Figure 2 Procedure for ontology design and evaluation

Another existing methodology is METHONTOLOGY, which defines a set of life cycle models and a development process to provide an overview of how an ontology should be developed. This methodology identifies the set of activities to be carried out during the development process. The life cycle models it proposed were the waterfall one, the incremental one (which ensures that each

version is compatible with the previous ones), and the one based on evolving prototypes (with essential similarities to agile development). Figure 3 shows an overview of the ontology life cycle and the activities proposed in this methodology.

In METHONTOLOGY, the main technical activities, as well as the evaluation and documentation activities, are considered in the LOT methodology.

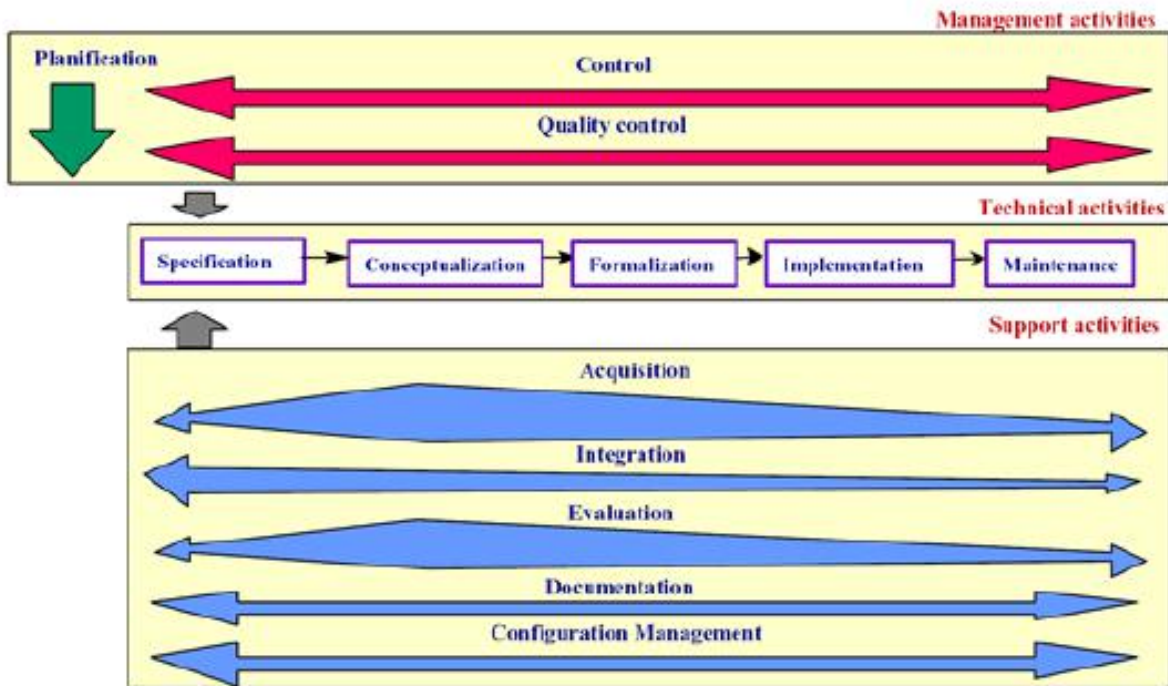


Figure 3 METHONTOLOGY development process and life cycle

The On-To-Knowledge methodology lies in the application-oriented development of ontologies. As shown in Figure 4, it includes five phases, namely: (a) feasibility study, (b) ontology start, (c) refinement, (d) evaluation, and (e) maintenance.

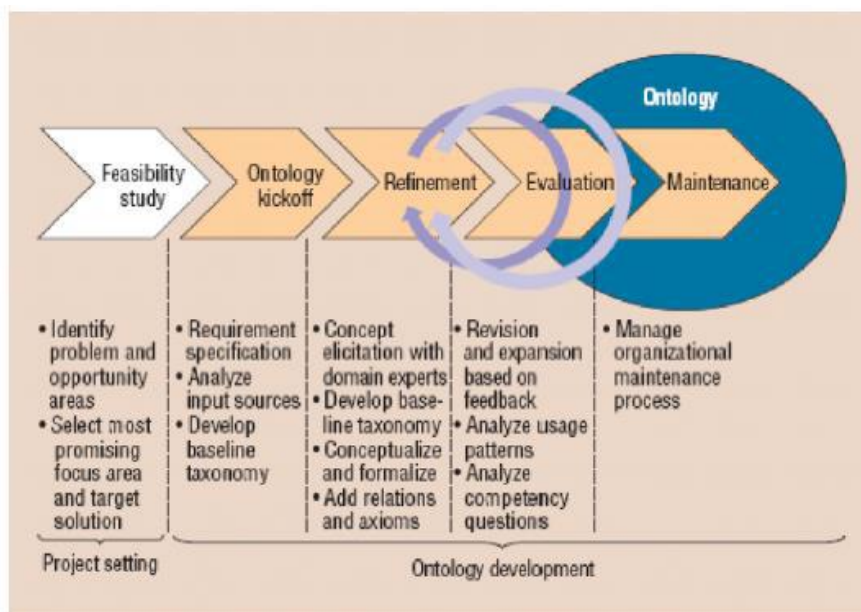


Figure 4 On-To-Knowledge development process and life cycle

During the ontology kick-off phase, ontology engineers should collect user requirements written as competency questions to provide an overview of possible queries to the system that can indicate the scope and content of the domain ontology. Next, during the evaluation phase, ontology engineers verify whether the target ontology satisfies the ontology requirements specification document and whether the ontology supports or answers the competency questions analysed in the kick-off phase of the project. During this evaluation phase, new requirements can arise that should be handled by the ontology. The LOT methodology considers refinement cycles in different phases inspired by this methodology.

Concerning the DILIGENT methodology, it was proposed to support ontology development in a distributed environment. In this scenario, the actors involved in the development of the same ontology have different complement skills, including ontology users and ontology developers. The general process proposed in this methodology, shown in Figure 5 includes five activities, namely (1) build, (2) local adaptation, (3) analysis, (4) revision and (5) local update. The distributed edition of ontologies is considered in LOT to be addressed in a technical way.

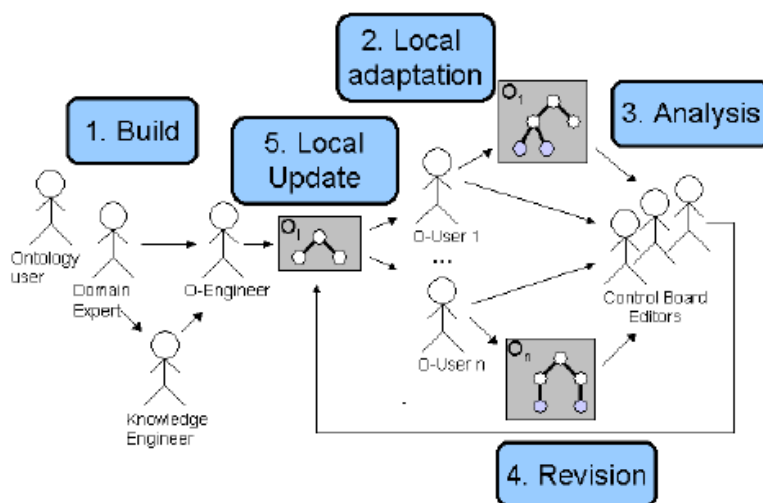


Figure 5 DILIGENT life cycle

The NeOn methodology was a combined European effort to provide precise guidelines, supported by an integrated development environment (the NeOn Toolkit), to develop network ontologies. The main goal of this methodology is to provide support for the collaborative development of ontologies and concrete guidelines for the reuse and reengineering of knowledge sources. To do so, this methodology identifies nine scenarios for building ontology networks, which are the most common during ontology network development. An overview of all these scenarios is shown in Figure 6.

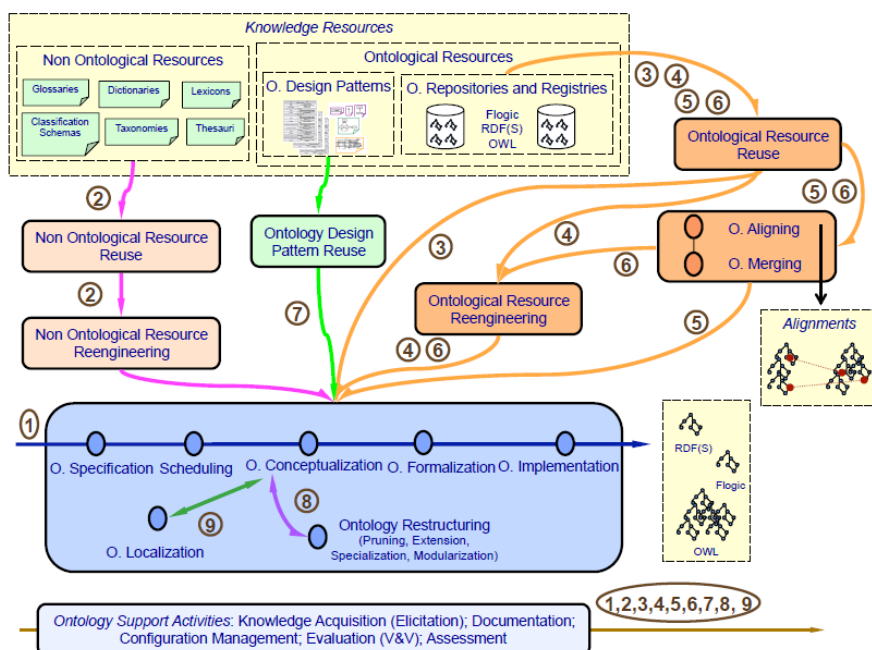


Figure 6 NeOn scenarios for building ontology networks

As shown in Figure 6, the methodology supports ontology implementation activities, such as specification, conceptualisation, and formalisation, as well as ontology support activities, such as documentation and evaluation. The LOT methodology considers the reuse of ontologies and non-ontological resources, reusing the NeOn methodology when applicable, for example as the NeOn methodology does not consider the publication activities there are no applicable guidelines for such step. In other cases, the guidelines have been adapted to more practical recommendations as for example the requirements elicitation.

RapidOWL methodology is based on the idea of iterative refinement, annotation, and structuring of a knowledge base through small incremental changes from multiple contributors. RapidOWL, which is based on agile methodologies, proposes a set of values from which a set of principles and practices are derived to establish those principles. The list of values, principles, and practices is shown in Figure 7. Some practices are followed by the LOT methodology as the short releases, view generation, joint ontology design, etc

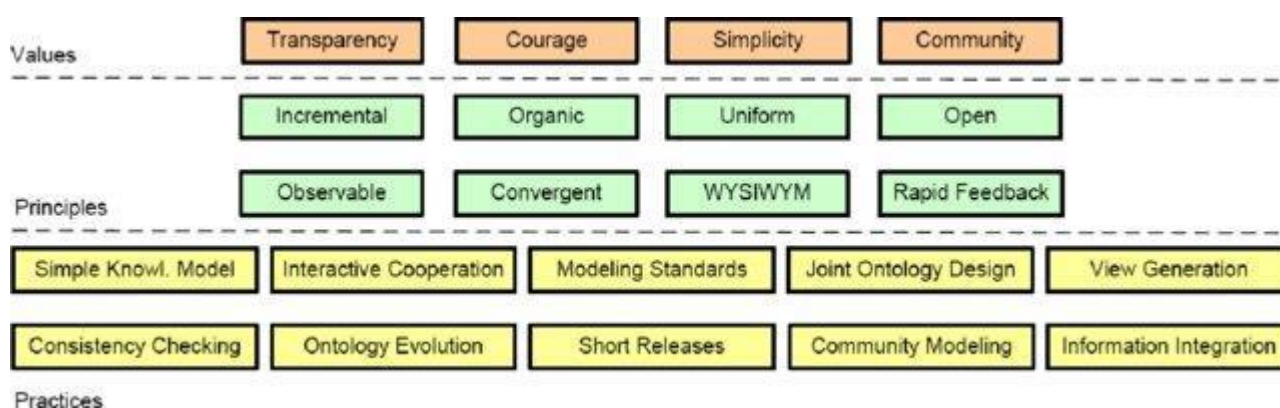


Figure 7 The building blocks of RapidOWL: Values, Principles, and Practices

It should be mentioned that RapidOWL does not prescribe a sequence of modelling activities that should be followed precisely.

Next, the AMOD is an agile methodology that adapt agile principles from software engineering into the development of ontologies. Consequently, it enables incremental and iterative into an ontology development. AMOD includes three phases (Figure 8):

- The pre-game phase: including the identification of the ontology goal and scope, tools and techniques, competency questions and available sources.
- The development phase: incorporating multiple and iterative cycles that are called sprints.
- The post-game: preparing for a final ontology.

The primary roles considered in AMOD are ontology owner (person representing customer needs to the ontology engineers), ontology engineer (person responsible for the implementation of the ontology) and ontology user (person reusing for a specific purpose the final ontology).

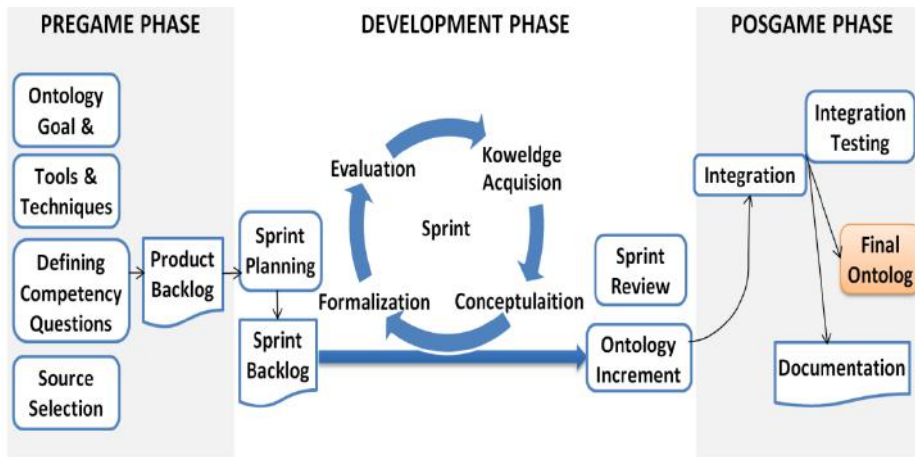


Figure 8 AMOD Framework

Finally, SAMOD is an agile methodology for the development of ontologies by means of small steps of an iterative workflow that focuses on creating well-developed and documented models. It includes three steps which are summarised in Figure 9. It should be noted that some practices of the SAMOD methodology as the information collection about specific domain is aligned with LOT and other methodologies, and also the process about modellet generation could be integrated in LOT development process if chosen by the developers.

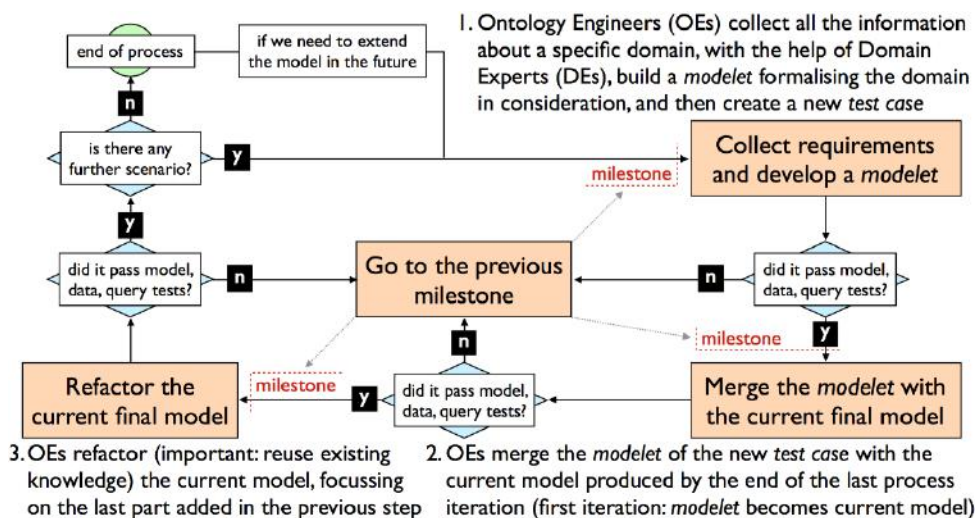


Figure 9 Brief summary of SAMOD

1. To define a new test case. Given a motivating scenario, ontology engineers and domain experts should produce a set of informal competency questions. Then, ontology engineers should create the modelet according to the motivating scenario, which is a stand-alone model describing a particular aspect of the domain. To create this modelet, ontology engineers can use a graphical representation written in a proper visual language, such as UML, to convert it automatically to OWL. Additionally, competency questions are translated into SPARQL queries to verify whether the modelet covers the related motivating scenario. Finally, ontology engineers should create an exemplar dataset that formalises all examples introduced in the motivating scenario according to the modelet.
2. To merge the current model with the modelet. At this stage, ontology engineers should merge the modelet with the current model, i.e., the version of the final model released at the end of the previous iteration. If there is a failure of any test, ontology engineers should go back to a previous milestone to solve the problem.
3. To refactor the current model. In the last step, ontology engineers work to refactor the current model by reusing existing knowledge, documenting it, and enriching the current model by using all the capabilities offered by OWL 2 to automatically infer as much information as possible starting from a small set of real data.

Each iteration of SAMOD aims to produce a new test case that will be added to the bag of test cases. Each test case describes a particular aspect of the same model, that is, the current model under consideration after one iteration of the methodology.

Here, we propose an analysis of the state-of-the-art methodologies to determine the “Mrs right” methodology that we must select for our work. Hence, we define 8 criteria that are detailed hereafter.

- **C1: Completeness.** Does the methodology support or include all common ontology development life cycle activities (specification, conceptualization, formalization, implementation, maintenance, knowledge acquisition, evaluation, documentation, publication¹)?
- **C2: FAIRness.** Does the methodology follow the FAIR principles?
- **C3: Reusability.** Is the methodology rooted to some well-established existing methodologies?
- **C4: Collaborative.** Does the methodology support collaborative development?
- **C5: Applicability.** Is the methodology domain independent?
- **C6: Agility.** Is the methodology aligned to agile approach?
- **C7: Modularity.** Is the methodology enabling modular ontology development?

¹ Publication following the 5 stars rules <https://bvatant.blogspot.com/2012/02/is-your-linked-data-vocabulary-5-star-9588.html>

Table 1 presents a detailed comparison of methodologies based on the established criteria. The analysis shows that the LOT methodology is the unique one covering all the criteria among the described methodologies. The other methodologies are not mature enough; mainly, most of them are missing some re-engineering aspects of ontology related to documentation and publication and omitting agility aspects. We noticed that most of the analysed methodologies provide few details about modularity and that no methodology is following the FAIR principles except the LOT.

Table 1 Comparison of the state-of-the-art methodologies: "yes" indicates that the methodology totally covers the criterion, and "no" indicates that the methodology does not totally cover the criterion. We mention "n/a" when the methodology does not provide information about this criterion.

	Methodology	C1	C2	C3	C4	C5	C6	C7
1	Grüninger & Fox	No	n/a	n/a	No	Yes	n/a	n/a
2	METHONTOLOGY	Yes	n/a	Yes	No	Yes	n/a	Yes
3	On-To-Knowledge	Yes	n/a	Yes	No	Yes	n/a	n/a
4	DILIGENT	No	n/a	No	Yes	Yes	No	n/a
5	NeOn	Yes	n/a	Yes	Yes	Yes	No	Yes
6	RapidOWL	No	n/a	No	Yes	Yes	Yes	n/a
7	SAMOD	Yes	No	Yes	Yes	Yes	Yes	Yes
8	AMOD	No	n/a	No	No	Yes	Yes	n/a
9	LOT	Yes	Yes	Yes	Yes	Yes	Yes	Yes

3. Input analysis

The LOT methodology was proposed and will evolve considering several inputs, which are presented in the following sections. These inputs include the results of: 1) feedback from the activities performed regarding domain ontologies; 2) ontology landscape collected; 3) focussed workshop organised; and 4) definition of development processes and requirements.

3.1 Domain ontologies development

One of the tasks is to review and formalise common requirements for ontology specifications and exploitation of each domain present in OntoCommons. To that end, expert group meetings are being organised to collect input on their requirements and needs in terms of ontological perspectives from stakeholders. Such requirements include specifications of each domain ontology, the core capabilities (e.g., competency questions) and specific perspectives of exploitation of the domain ontology.

Feedback from this task is considered in the LOT methodology to support the needs identified in the expert group meetings, such as the demand for specific guidelines for the ontology requirements specification.

3.2 Ontology landscape analysis

The purpose of this OntoCommons survey was to collect more descriptive and technical metadata about existing ontologies and was openly available to be answered by anyone with some knowledge in ontologies. The questions included in this survey are those shown in **Error! Reference source not found.**

Responses to this survey will be considered as input to the ontology reuse activity. Furthermore, additional best practices discovered from survey responses could be integrated into the methodology for specific domains.

3.3 First focused workshop on domain ontologies

The *Report on the first focused workshop on domain ontologies* describes the results obtained from the workshop organised on June 7th, 2021, co-located with the 18th European Semantic Web Conference (ESWC) to collect feedback from domain ontologies for research data management in industry commons of materials and manufacturing. The discussions and gaps collected from the event were also considered for the definition of the methodology and associated guidelines. Some of the highlights obtained were the following:

- *'We need to give industry some useful solutions. How do we make it a more cost-effective process (quicker, cheaper and more effective)?'*
- *'More involvement from industry is needed.'*
- *'Templates must be simple and ontology patterns can really work well. Web forms should be available to the domain experts. It was found that experienced domain experts with no semantic background could contribute in this way very effectively and quite independently to ontology building.'*
- *'We really need to have both domain experts and ontologists to meet'*

3.4 Demonstrators

The report on 'Requirements on ontology tools and ontologies and criteria for selection of further cases' provides a set of requirements on ontology tools and ontologies for the demonstrators in OntoCommons, also including the need for FAIRness for the data in the demonstrators.

This set of requirements is divided into several topics, including application of ontologies, standardisation, development, and extension of ontologies and tools. The LOT methodology aims at

addressing the requirements that deal with the methodological perspective in ontology engineering. Table 2 and Table 3 show a subset of the requirements defined together with how they are covered in the LOT methodology. These tables indicate in the column 'covered' whether the requirements are covered by the methodology (✓) or whether LOT will provide some resource to partially cover it (≈).

Table 2 requirements- Application of ontologies

UID	Title	Solution in LOT	Covered
CRQ_U_01	Support domain description	Guidelines for the definition of ontology requirements	✓
CRQ_U_02	Requirement traceability	Guidelines for the definition of ontology requirements	✓
CRQ_U_07	Allow for quality metrics	Definition of ontology metrics during the ontology evaluation activity	≈
CRQ_U_08	Ontology-based data access and correlation	Addition of a new activity in LOT for ontology usage, which includes ontology-based data access.	Out of scope
CRQ_U_09	Documentation of domains	Definition of guidelines and good practices for the generation of documentation. Tools recommended	✓
CRQ_U_14	Ontology reuse, harmonisation, and modularisation	Guidelines for ontology reuse in the reuse activity	✓
CRQ_U_15	Non-ontology expert user	Good practices for metadata included in the ontology and documentation	✓

Table 3 requirements- Development of ontologies

UID	Title	Solution by LOT	Covered
CRQ_D_02	Controllability	Recommendations and guidelines for ontology evaluation	✓
CRQ_D_03	Compatibility	Guidelines for ontology reuse in the reuse activity	✓
CRQ_D_04	Documentation for interoperability	Guidelines for ontology reuse in the reuse activity	✓

CRQ_D_05	Usability and understandability	Resources for the conceptual modelling	≈
CRQ_D_06	Ontology Scope	Guidelines for requirements definition and its verification	✓
CRQ_D_07	Methodology user audit	Good practices for metadata included in the ontology and documentation	✓
CRQ_D_08	Ontology reuse	Guidelines for ontology reuse in the reuse activity	✓
CRQ_D_09	Development of a methodology for ontologies	LOT defines activities, includes recommendations, and proposes tools	✓
CRQ_D_10	Methodology for ontology engineering	LOT defines activities, includes recommendations, and proposes tools	✓
CRQ_D_11	Methodology - conceptual phase	Resources for the conceptual modelling	≈

4.LOT Methodology

This section presents Linked Open Terms (LOT), an agile and iterative ontology development methodology that includes four activities: 1) ontology requirements specification, 2) ontology implementation, 3) ontology publication, and 4) ontology maintenance. The LOT methodology is based on the NeOn methodology [13] and it adapts previous ontology development processes from the European projects VICINITY [3], BIMERR [4], DELTA [5] and EasyTV [14] to the particularities of OntoCommons. Figure 10 shows an overview of the processes that must be performed and the product that results from them.

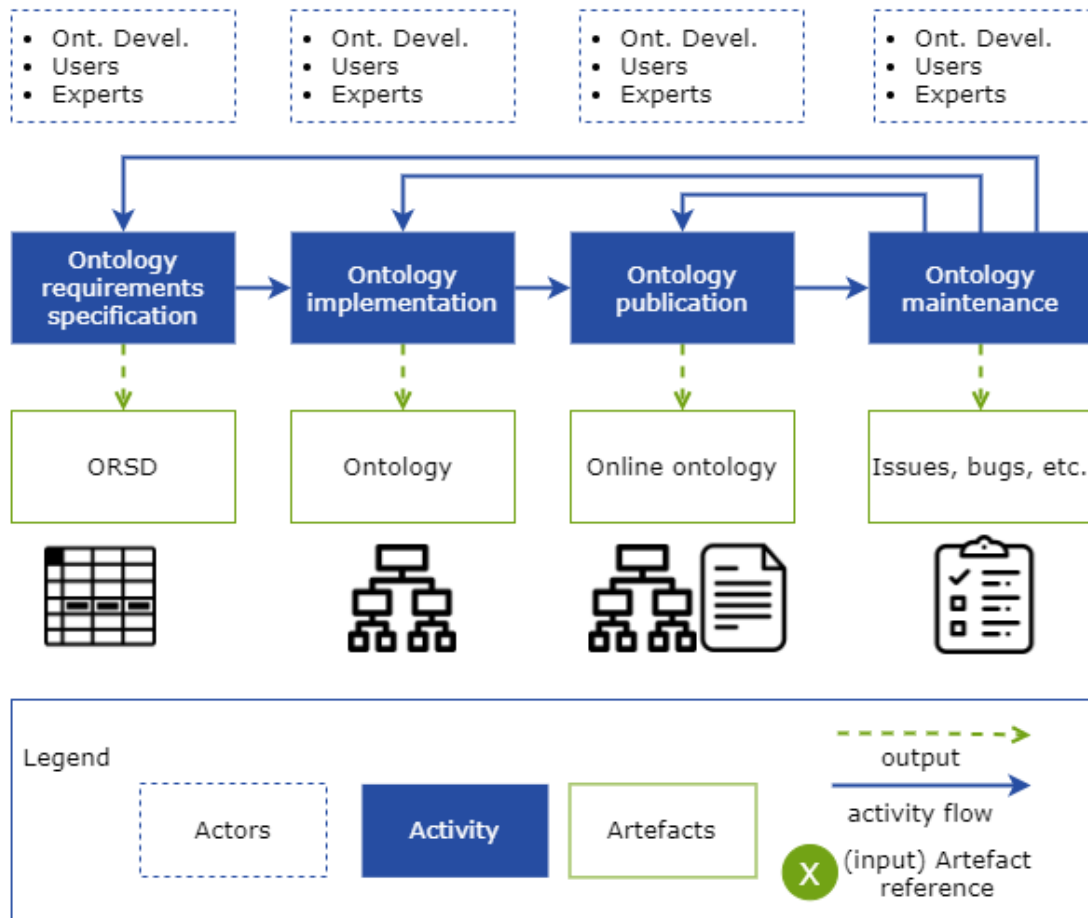


Figure 10 Overview of LOT activities

4.1 Ontology requirements specification activity

The ontology requirements specification activity aims to state why the ontology is being built and to identify and define the requirements the ontology should fulfil. In this step, the participation and commitment of experts in the specific domain at hand is required to generate the appropriate industry perspective and knowledge. An overview of the sub-activities to be performed during this first activity is shown in Figure 11 and described in the following subsections.

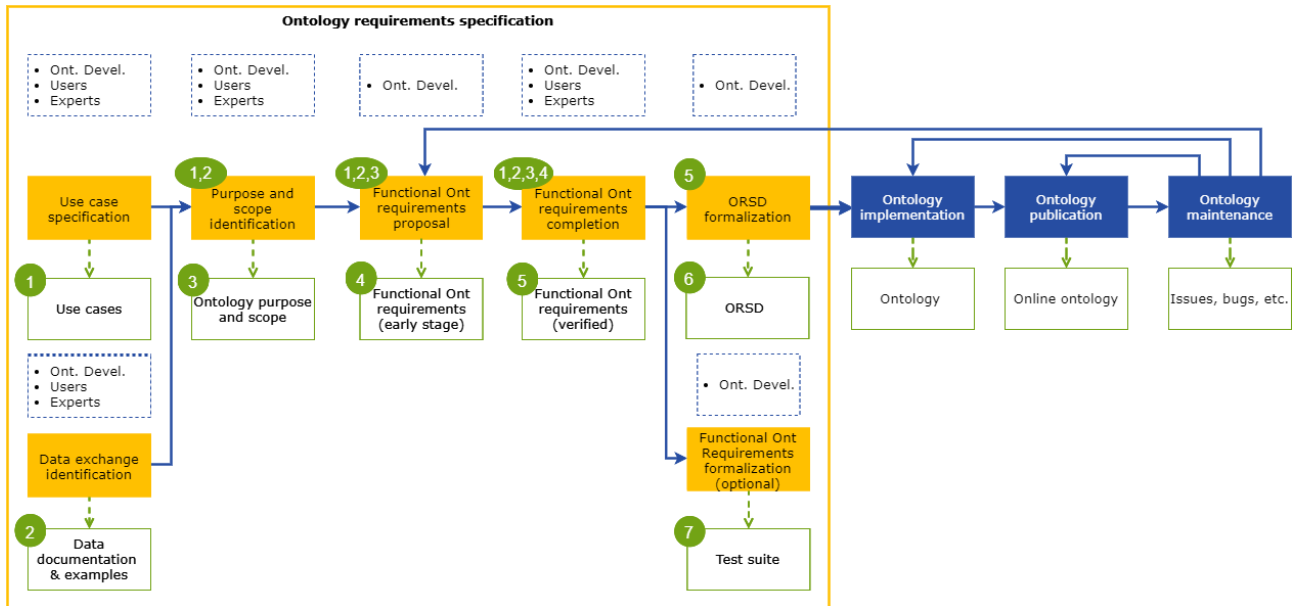


Figure 11 Ontology requirements specification sub-activities

4.1.1 Use case specification

The goal of this first activity is to provide a vision of the potential use that the ontology will have. This activity involves domain experts, users, and ontology engineers. The output of this activity is a list of use cases that describe situations that are desired to be reached with the data that are described by the ontology. Therefore, they guide the specification of the ontology requirements. An example of a use case is the following:

1. **Description:** *Finding parking spaces available in certain areas and times of day is an almost impossible task in certain cities. Faced with this problem, a city council has deployed a network of sensors in public car parks to obtain data in real time and to provide citizens with information about free and occupied places. In addition, it has placed display panels of available parking spaces on the streets and made available to the public a mobile application for guided parking. Thanks to this system, citizens can know in real time in which public parking there are free places to park their car.*
2. **Actor:** *citizen, application*
3. **Flow:** *Mike, while driving his car to the city centre, activates the parking application through a voice command. The application requests the destination address. Mike specifies the destination address to the application. The application, which has access to the GPS of his mobile, inspects the data emitted by the parking sensors, identifies the parking places closer to Mike's destination from its current location, and recommends him a route to park. Mike follows the route suggested by the application, arrives at the place and parks.*

4.1.2 *Data exchange identification*

The goal of the data exchange identification activity is to provide the ontology development team with the necessary documentation about the domain to be modelled. In this case, the documentation to be shared might correspond to:

- Datasets
- Regulations
- Standards
- Data formats
- Software manuals
- APIs specifications
- Database schemas, etc.

In this activity, ontology users (e.g., software developers) are responsible for providing this documentation to ontology engineers. The output of this activity is a set of domain documents and resources.

4.1.3 *Purpose and scope identification*

To identify the purpose and scope of the ontology, the ontology development team works in collaboration with users and domain experts. Communication between domain experts, users, and the ontology development team can be carried out through online or physical meetings. The output of this activity is a text document that describes the purpose and scope of the ontology.

4.1.4 *Functional ontology requirement proposal*

Considering the documentation and data provided in previous activities, the ontology development team, supported by users and domain experts, generates a first proposal of ontological requirements.

First, the ontology development team should check if there are existing ontology requirements that deal with the same domain so that they can be reused. Subsequently, new requirements should be proposed. Such requirements can be written in the form of:

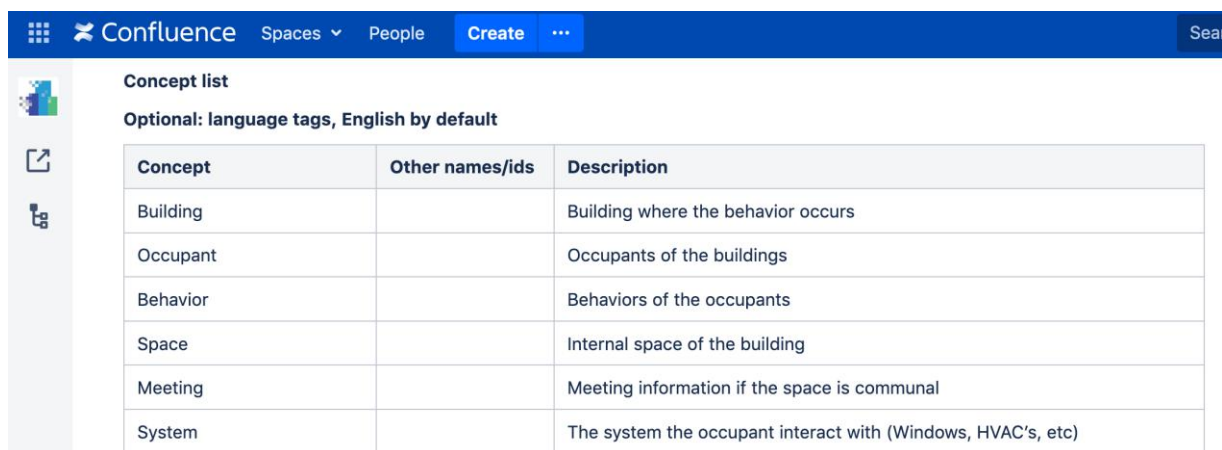
- **Competency questions.** A set of competency questions is proposed following the well-known technique proposed by [15] that suggests elaborating a set of questions that an ontology must be able to answer taking into consideration the purpose and motivation of building the ontology. Competency questions might be accompanied by answers or expected results. Examples of requirements written as competency questions are the following:
 - What are the relationships in which a partnership is involved?
 - What is the relation between organization and devices?
 - How many organizations can have a partnership?

- What are the parameters that have a service? Owner, avatar, description, name.
- **Natural language sentences.** If domain experts have no knowledge about ontology data generation and querying, we recommend writing the requirements in the form of natural language sentences. This technique is an alternative that could be used in combination with competency questions. Examples of requirements written as natural language sentences are the following:
 - A human user interacts with applications.
 - The digital user consumes services.
 - A physical entity is controlled by an actuator.
 - A thermometer is a type of sensor.

More examples of requirements written as natural language sentences can be found in the ontology portals developed in the VICINITY² and DELTA³ H2020 projects.

- **Tabular information.** This tabular technique was proposed in METHONTOLOGY [16] and consists of creating 3 types of tables:
 - Concepts
 - Relations
 - Attributes

Figure 12, Figure 13 and Figure 13 show examples of METHONTOLOGY tables for extracting requirements in which several ontologies about domains involved in energy efficiency for building renovation processes were developed, such as buildings, weather, sensors and actuators.



Concept	Other names/ids	Description
Building		Building where the behavior occurs
Occupant		Occupants of the buildings
Behavior		Behaviors of the occupants
Space		Internal space of the building
Meeting		Meeting information if the space is communal
System		The system the occupant interact with (Windows, HVAC's, etc)

Figure 12 'Concepts' for ontology requirements

² <http://vicinity.iot.linkeddata.es/vicinity/>

³ <http://delta.iot.linkeddata.es/>



Concept	Relation	Description	Target object (should appear in the "Concept list")	Max cardinality	Ordering Needed (applicable for concepts with Max cardinality over 1)	(ignore if not sure or not applicable) Other characteristics: symmetric, transitive, has some special behaviour or meaning? etc.	Needed by	standards
Building	hasSpace		Space					
Space	meeting		Meeting					
Space	hasSystem		System					
Space	usedBy		Occupant					

Figure 13 Table of 'Relations' for ontology requirements



Concept	Attribute	Description	Value type expected (integer, boolean, float, list of specific values, etc.)	Max cardinality	Ordering Needed (applicable for concepts with Max cardinality over 1)	Unit of measure (applicable only for "measure" data types)	Sensitive data (if applicable, e.g. personal data that need to be anonymized in BIF)	Timezone (applicable only for datetime data types)	Related standard (if applicable)
Space	description	Description of the space	String	1					obXML
Space	maxNumberOccupants	Maximum number of occupants	Integer	1					obXML
Space	minNumberOccupants	Minimum number of occupants	Integer	1					obXML
Meeting	meetingDuration	Duration of meeting	Integer	1		seconds			obXML

Figure 13 Table of attributes for ontology requirements

- **MODA:** This tabular technique is based on the MODA specification⁴ and is aligned with the tabular techniques proposed in METHONTOLOGY [16]. This technique also consists of 3 types of tables:
 - Concepts
 - Relations
 - Attributes

Figure 14, Figure 15 and Figure 16 show examples of MODA tables for extracting ontology requirements in the materials domain.

⁴ <https://emmc.eu/resources/moda/>

	A	B
1	Material entity	Description
2	Atom	The smallest particle of a chemical element that can exist.
3	Electron	A stable subatomic particle with a charge of negative electricity, found in all atoms and acting as the primary carrier of electricity in solids
4		
5		

Figure 14 'Concepts' to extract ontology requirements from MODA specifications

	A	B	C	D	E	F
1	Material entity	Material Relation	Target entity	Description	Maximum cardinality	Any comment regarding more restrictions of the relation
2	Electron	isRepresentedAs	MaterialPoint		1	
3						
4						
5						
6						
7						

Figure 15 Table of 'Relations' to extract ontology requirements from MODA specifications

	A	B	C	D	E
1	Material entity	Physical quantities/ Attributes of the material entity	Description	Value type expected (integer, string, etc.)	Units of measure (seconds, degrees, etc.)
2	Electron	hasElectronDensity	Relationship specifying the density of an electron	float	
3	Electron	hasElectronPotential	Relationship specifying the pseudo potentials for the implicit core electron	float	
4					
5					

Figure 16 Table of 'Attributes' to extract ontology requirements from MODA specifications

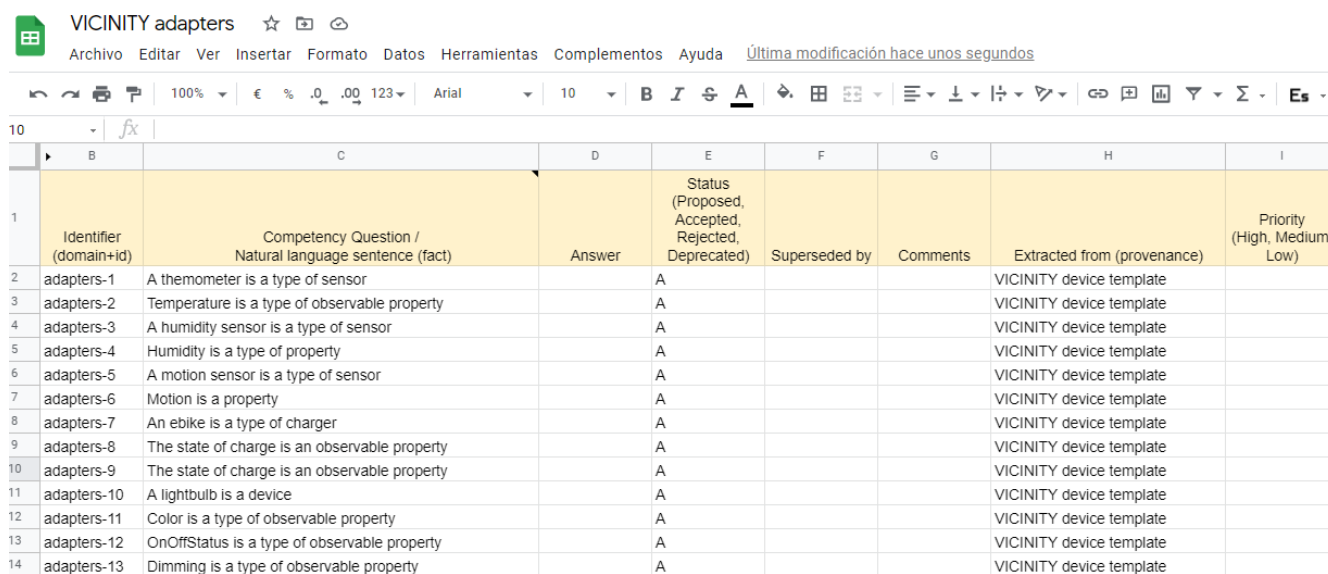
We recommend using the METHONTOLOGY tables if the data is structured, if the ontology development have APIs, or if there is a close collaboration with software developers. Moreover, we recommend using MODA tables if MODA is used for documenting materials modelling workflows.

If the set of requirements is written in the form of competency questions or natural language sentences, it can also be stored following a tabular approach and include additional information. We recommend including at least the following fields:

- Requirement identifier, which needs to be unique for each requirement.
- The domain of the requirement (Domain Industrial Ontologies).
- The competency question or a natural language sentence.
- The answer to the competency question

- Status of the requirement, which can be: (1) Proposed, (2) Accepted, (3) Rejected, (4) Ongoing, or (5) deprecated.
- In case the requirement is deprecated, the identifier of the updated requirement is used.
- Comments on the requirement.
- Provenance of the requirement (e.g., if it is reused).
- Priority of the requirement, which can be: (1) High, (2) Medium, or (3) Low.

An example of the proposed requirements is shown in Figure 17. We also provide a spreadsheet that can be used as the template to write such requirements;⁵ the set of requirements associated with the same domain should be stored in the same document to facilitate its reusability.



Identifier (domain+id)	Competency Question / Natural language sentence (fact)	Answer	Status (Proposed, Accepted, Rejected, Deprecated)	Superseded by	Comments	Extracted from (provenance)	Priority (High, Medium, Low)
adapters-1	A thermometer is a type of sensor		A			VICINITY device template	
adapters-2	Temperature is a type of observable property		A			VICINITY device template	
adapters-3	A humidity sensor is a type of sensor		A			VICINITY device template	
adapters-4	Humidity is a type of property		A			VICINITY device template	
adapters-5	A motion sensor is a type of sensor		A			VICINITY device template	
adapters-6	Motion is a property		A			VICINITY device template	
adapters-7	An ebike is a type of charger		A			VICINITY device template	
adapters-8	The state of charge is an observable property		A			VICINITY device template	
adapters-9	The state of charge is an observable property		A			VICINITY device template	
adapters-10	A lightbulb is a device		A			VICINITY device template	
adapters-11	Color is a type of observable property		A			VICINITY device template	
adapters-12	OnOffStatus is a type of observable property		A			VICINITY device template	
adapters-13	Dimming is a type of observable property		A			VICINITY device template	

Figure 17 Excerpt of ontology requirements

To help in the definition of requirements, the CORAL Corpus can be used⁶. CORAL includes a dictionary of lexico-syntactic and a set of 834 requirements extracted from real-world ontologies that are annotated according to their lexico-syntactic patterns. The dictionary of patterns identifies different types of ontology requirements and how they are specified, for example, a requirement that has the pattern 'What is NP<class>?', asks about the existence of a class in the ontology named NP<class>. This dictionary of patterns presents a set of ambiguous expressions that can lead to multiple implementations in the ontology; for example, the use of the verb 'to have' in a requirement can be translated both as a datatype property and an object property in the ontology. These ambiguous expressions should be avoided from the requirements specification, since they hinder

⁵ https://docs.google.com/spreadsheets/d/1cZ5dlmRv8WtHE_Q-4myjlPh7d_hJTGt-ClS3J0avsE/edit?usp=sharing

⁶ <http://coralcorpus.linkeddata.es/>

the translation from the requirement to the ontology. The set of 834 annotated requirements can also be taken as a reference for the definition of ontology requirements.

4.1.5 *Functional ontology requirement completion*

During this activity, domain experts and users in collaboration with the ontology development team validate whether the ontology requirements defined in the previous step are correct and complete. The following criteria can be used in this validation task as stated in [15]:

- A set of requirements is correct if each requirement refers to some feature of the ontology to be developed.
- A set of requirements can be considered complete if users and domain experts review the requirements and confirm that they are not aware of additional requirements.
- A set of requirements can be considered internally consistent if there are no conflicts between them.
- A set of requirements is verifiable if there is a finite process with reasonable cost that tests whether the final ontology satisfies each requirement.
- Each requirement must be understandable to end-users and domain experts.
- An ontology requirement is unambiguous if it has only one meaning; that is, if it does not admit any doubt or misunderstanding.
- A set of requirements is concise if every requirement is relevant and if there are no duplicated or irrelevant requirements.
- A set of requirements is realistic if every requirement meaning makes sense in the domain.
- A set of requirements is modifiable if its structure and style allow one to change issues in an easy, complete, and consistent way.

4.1.6 *ORSD formalisation*

Once the ontology development team has all the information about the requirements, they create the Ontology Requirements Specification Document (ORSD) [17]. This specification document stores all the functional and non-functional requirements identified and the information associated with them.

The LOT methodology proposes a template for the ORSD, which is stored in a GitHub repository to be used by users⁷. This template is an adaptation of the ORSD proposed in the NeOn methodology [13]. However, not all fields included in the NeOn methodology are included in this ORSD template,

⁷ <https://github.com/oeg-upm/LOT-resources>

they have been marked as optional as it has been observed that some fields are not relevant for the ontology implementation phase, so it is up to the developers to spend effort in filling them in. An example of ORSD is shown in Figure 18.

If the ontology is published on the Web, it would be advisable to make it available with the requirements that lead to the obtained model. For example, the ontology portals developed in the VICINITY⁸ and DELTA⁹ H2020 projects store all the resources associated with the ontologies, including the requirements. Moreover, the requirements could also be included in the GitHub repository where the ontology is stored. An example of an ontology GitHub repository is the one associated with the Web of Things ontology.¹⁰

Ontology Requirements Specification Document	
1	Purpose
2	Scope
3	Implementation Language (optional)
4	Intended End-Users (optional)
5	Intended Uses
6	Ontology Requirements
	a. Non-Functional Requirements
	b. Functional Requirements: Lists or tables of requirements written as Competency Questions and sentences
7	Pre-Glossary of Terms (optional)
	a. Terms from Competency Questions
	b. Terms from Answers
	c. Objects

Figure 18 Template of the ORSD

⁸ <http://vicinity.iot.linkeddata.es/vicinity/>

⁹ <http://delta.iot.linkeddata.es/>

¹⁰ <https://github.com/mariapoveda/wot-ontology>

4.1.7 Functional ontology requirements formalisation

In this optional activity, the ontological requirements written in natural language are formalised into test cases.

These test cases should include the identifier of the associated requirement, the description of the test case (which includes a link to the ORSD), and the SPARQL queries extracted from the competency question together with the expected result of the query. Tests can be defined as SPARQL queries or as test expressions¹¹. In addition, requirements and tests can be stored in RDF files following the Verification Test Case ontology¹². SPARQL queries and tests can be executed on the ontology to verify if the ontology satisfies the ontological requirements identified later in the ontology development process during the ontology evaluation activity.

```

:Test-case1 a vtc:TestCaseDesign, owl:NamedIndividual ;
  vtc:isRelatedToRequirement <http://vicinity.iot.linkeddata.es/vicinity/requirements/report-core.html#platform2>;
  dc:description "What is an IoT device?" ;
  vtc:desiredBehaviour "Device subclassOf PhysicalThing" .

:Test-case2 a vtc:TestCaseDesign, owl:NamedIndividual ;
  vtc:isRelatedToRequirement <http://vicinity.iot.linkeddata.es/vicinity/requirements/report-core.html#platform4>;
  dc:description "What is a partnership?" ;
  vtc:desiredBehaviour "Organization subclassOf SymmetricProperty(hasPartnershipWith) some Organization" .

:Test-case3 a vtc:TestCaseDesign, owl:NamedIndividual ;
  vtc:isRelatedToRequirement <http://vicinity.iot.linkeddata.es/vicinity/requirements/report-core.html#platform5>;
  dc:description "What attributes has a partnership?" ;
  vtc:desiredBehaviour "Organization subclassOf SymmetricProperty(hasPartnershipWith) some Organization" .

:Test-case4 a vtc:TestCaseDesign, owl:NamedIndividual ;
  vtc:isRelatedToRequirement <http://vicinity.iot.linkeddata.es/vicinity/requirements/report-core.html#platform8>;
  dc:description "Which are the relationships a partnership is involved in?";
  vtc:desiredBehaviour "Organization subclassOf hasPartnershipWith some Organization".

:Test-case5 a vtc:TestCaseDesign, owl:NamedIndividual ;
  vtc:isRelatedToRequirement <http://vicinity.iot.linkeddata.es/vicinity/requirements/report-core.html#platform9>;
  dc:description "How many organizations can have a partnership?";
  vtc:desiredBehaviour "Organization subclassOf SymmetricProperty(hasPartnershipWith) some Organization" .
    
```

Figure 19 Example of tests using the Verification Test Case ontology

4.2 Ontology implementation

The goal of the ontology implementation activity is to build the ontology using a formal language, based on the ontological requirements identified by the domain experts. After the first set of requirements is defined, the ontology implementation phase is carried out. Ontology developers

¹¹ <http://themis.linkeddata.es/tests-info.html>

¹² <https://w3id.org/def/vtc#>

schedule and plan the development of the ontology according to the prioritization of requirements in the ontology requirements specification activity, if needed. The ontology development team builds the ontology iteratively, implementing a certain number of requirements in each iteration. The output of each iteration is a new version of the ontology. Figure 20 shows an overview of the sub-activities to be performed during the ontology implementation.

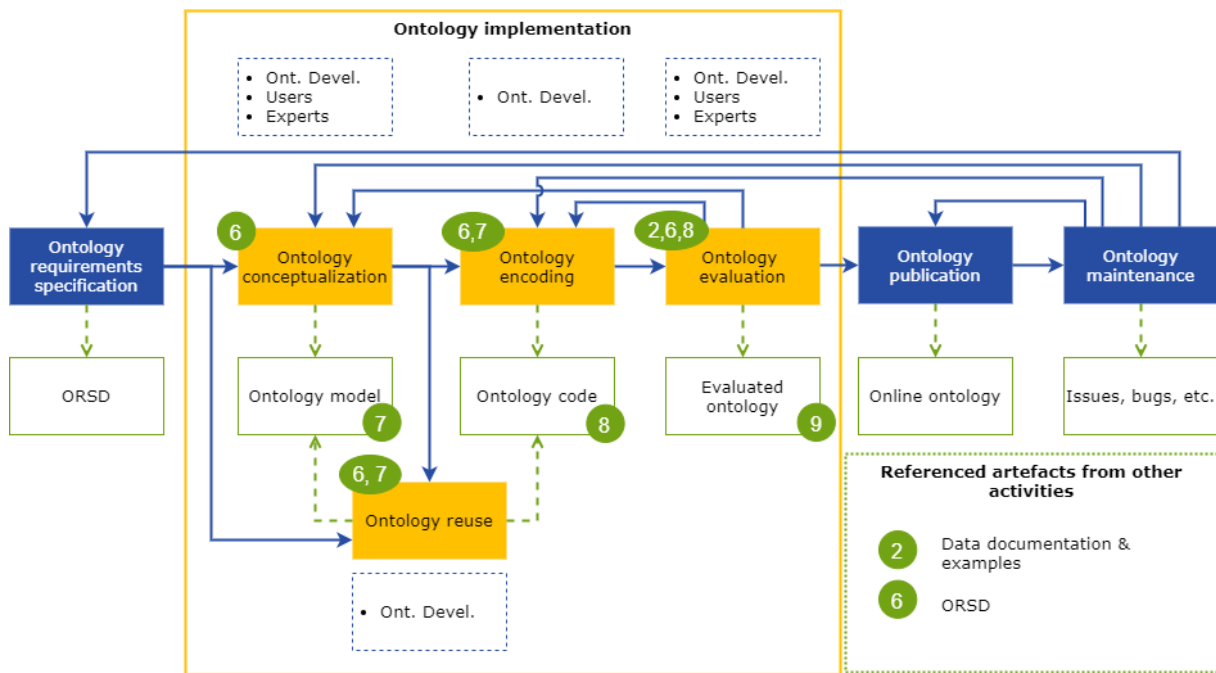


Figure 20 Ontology implementation subactivities

4.2.1 Ontology conceptualisation

The aim of this activity is to build an ontology model from the ontological requirements identified in the requirements specification process that represents the domain of the ontology. Therefore, during the conceptualisation of the ontology, the domain knowledge obtained from the previous activity is organized and structured into a model by the ontology development team. This conceptualization usually does not include all the constraints that a formal language imposes, and the level of detail is decided by the development team.

To perform this conceptualization, the Chowlk notation¹³, which is based on UML¹⁴, or UML_Ont [18] can be used, as well as other systems as OWLGrEd¹⁵, diagramming tools as MS Visio or draw.io, as well as non-digital tools as pen and paper or a blackboard.

¹³ https://chowlk.linkeddata.es/chowlk_spec

¹⁴ <https://www.uml.org/>

¹⁵ <http://owlgred.lumii.lv/>

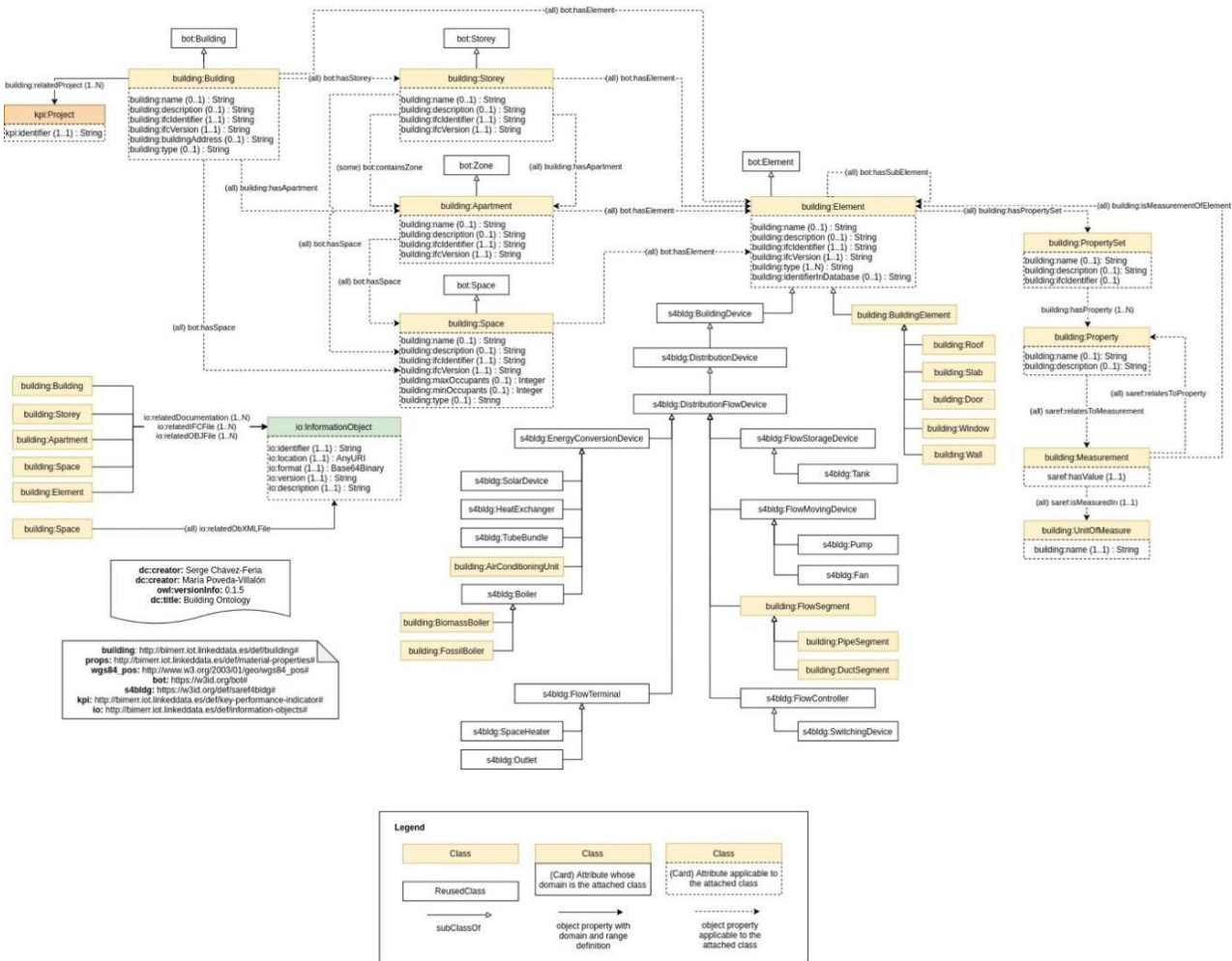


Figure 21 Example of conceptualisation using the Chowlk notation

There are existing guidelines that help this conceptualisation step, such as the Ontology Development 101¹⁶ or the W3C Semantic Web Best Practices and Deployment Working Group¹⁷.

4.2.2 Ontology encoding

The purpose of the encoding activity is to implement the ontology in an implementation language, such as OWL. The ontology code resulting from this activity includes, in addition to the ontology classes, properties, and axioms. Furthermore, following the FAIR principles [19], the ontology code should also include ontology metadata, such as creator, title, publisher, license, and version of the ontology in addition to metadata for each of the ontology terms, i.e., classes and properties.

¹⁶ <http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html>

¹⁷ <https://www.w3.org/2001/sw/BestPractices/>

To manage the ontology versions, the following version convention, which is based on Software Engineering¹⁸, was adopted. Following this convention, each release will follow the pattern **v.major.minor.fix**, where each field follows the rules:

- **major**: The field is updated when the ontology covers the complete domain it intends to model. That is, it is a complete product and covers the final goal of the development.
- **minor**: The field is updated when:
 - All the requirements of a subdomain are covered.
 - Documentation is added to the ontology.
- **fix**: The field is updated when:
 - Typos or bugs are corrected in the ontology.
 - Classes, relationships, axioms, individuals, or annotations are added, deleted, or modified, but the domain is not covered.

In each iteration, the minor and fix fields might be changed from zero to several times.

4.2.3 *Ontology reuse*

Ontology reuse refers to the activity of using available ontological resources to solve different problems [13]. There are two types of reuse:

- **Hard reuse**: This type of reuse is implemented by the OWL construct `owl:imports`, and the ontology is reused as a whole. In this case, the imported ontologies will also be part of the ontology being built.
- **Soft reuse**: This type of reuse is implemented by referring to other URIs of ontology elements. It could also include the reproduction of some parts of the reused ontology in the ontology being built.

In addition, Ontology Design Patterns can also be used to reuse ontological resources according to the NeOn methodology. There is an Ontology Design Patterns portal that provides a catalogue of patterns that can be used. Guidelines on the reuse of ontology can be found in [20] and [21].

The reuse activity can be performed during the conceptualization or the encoding activities. If the ontology development team knows about existing ontologies in the domain, such as standards or well-known ontologies, the developers can drive the conceptualisation based on them. However, we discourage looking for ontologies exhaustively from the ontology requirements specification activity without having a clearer idea of the conceptualisation.

¹⁸ <https://semver.org>

4.2.4 *Ontology evaluation*

This activity refers to the verification of the technical quality of an ontology against a reference frame [13], which should be performed before its online publication. The ontology development needs to consider different aspects, including:

- The logical consistency of the ontology.
- The detection of bad practices (modelling mistakes, lack of (FAIR) metadata, etc.)
- The detection of syntactic, modelling, or semantic errors.
- The coverage of the requirements scheduled for the ontology, to ensure that the ontology is completed regarding the domain experts needs.

The use of SPARQL queries to gather data based on ontology requirements or the use test expressions to verify the ontology using the ontology requirements usually helps domain experts and users to identify misunderstandings and missing knowledge in the ontology at an early stage of the development. The use of data coverage analysis is a useful practice if the ontology to be developed is part of a software product or if only ontology developers are involved in the ontology development process.

- The data coverage regarding the domain the ontology is modelling.

4.3 Ontology publication

During the ontology publication activity, the ontology development team provides an online ontology that is accessible both as a human-readable document and as a machine-readable file from its URI. Figure 22 shows an overview of the sub-activities to be performed during the publication of the ontology, which should follow the recommendations for providing FAIR ontologies [19].

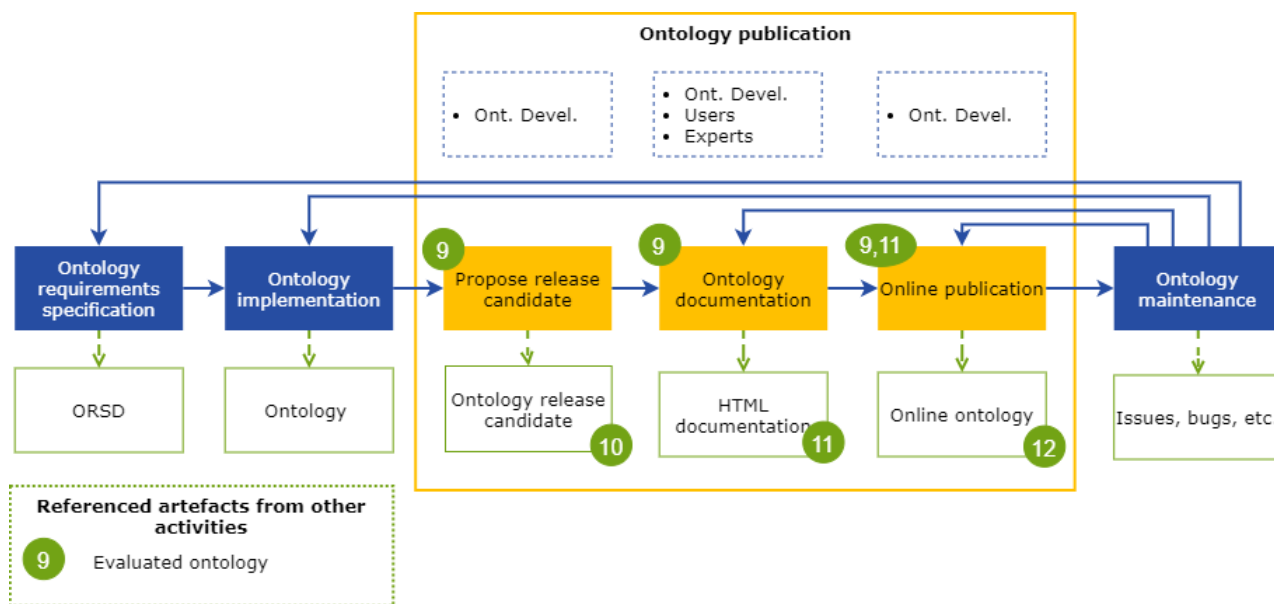


Figure 22 Ontology publication subactivities

4.3.1 Documentation generation

Taking as input the ontology generated in the previous activities, the ontology development team should generate the ontology documentation. Following FAIR practices [19], this documentation should include the following.

- An HTML description of the ontology which describes the classes, properties, and data properties of the ontology. Domain experts must collaborate with the ontology development team to describe the classes and properties.
- Metadata associated with the ontology, such as the license URI and title that were used. These metadata may also include the creator, publisher, date of creation, last modification, and version number.
- Diagrams that store the graphical representation of the ontology, including taxonomy and class diagrams.
- Custom diagrams with examples that illustrate how to use ontologies in practice.
- The documentation should also include links to different formats of serialization of the ontology, such as TTL, JSON-LD or RDF/XML.

We recommend including graphical representations of how to instantiate the ontology and some examples of use in the HTML documentation to allow domain experts and users to better understand the ontology and how to use it.

Since after the evaluation activity the ontology may change, we recommend generating the documentation after the evaluation activity, although small parts of the ontology documentation can be generated during the conceptualisation and encoding activity (e.g., adding metadata to the ontology).

4.3.2 *Online publication*

During this activity, the ontology, which should already be validated and documented, is published on the Web. The ontology should be accessible through its URI as a machine-readable and human-readable file by using content negotiation. This way, the ontology resolves to its HTML documentation when accessed by a user in a browser; and it resolves to an RDF serialization when loading it into an ontology editor [19].

4.4 Ontology maintenance

During this activity, the ontology is updated and new requirements can be proposed to be added to the ontology. Moreover, during this activity, the ontology development team, together with domain experts and users, can identify and correct errors in the ontology. Figure 23 shows an overview of the sub-activities to be performed during ontology maintenance.

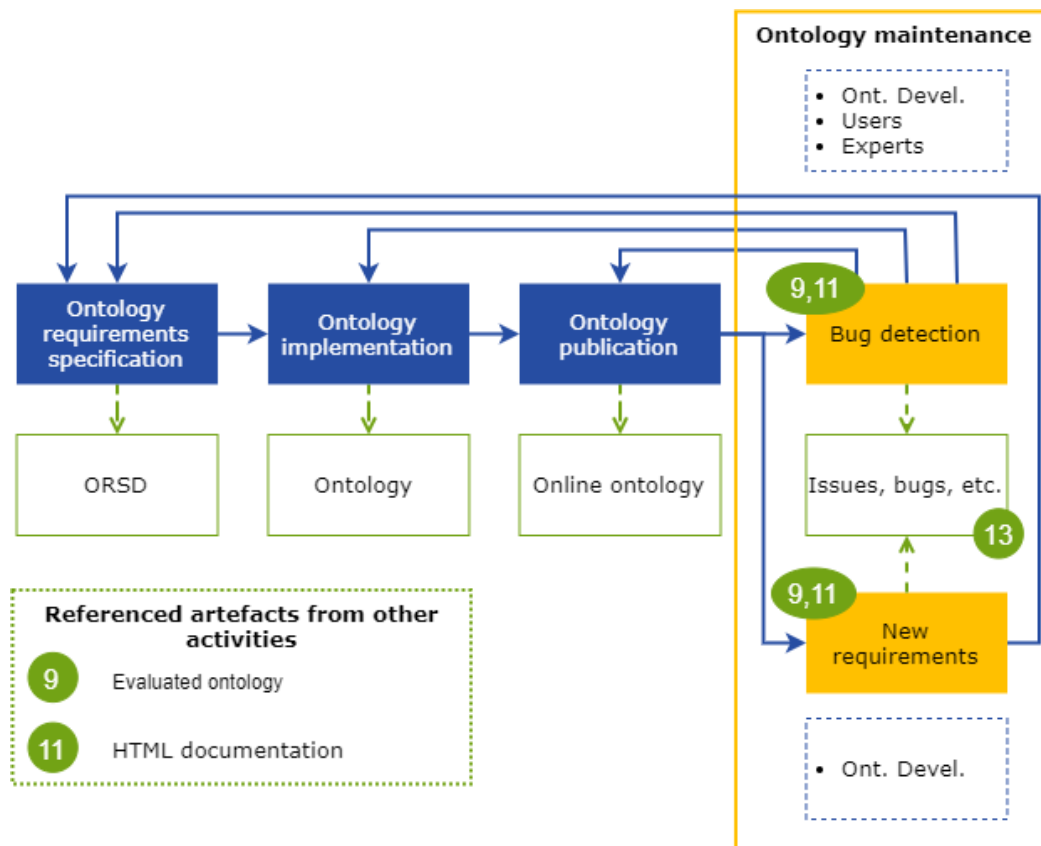


Figure 23 Ontology maintenance sub-activities

4.4.1 Bug detection

Once the ontology developers have published the ontology, any user, developer, or domain expert can detect and inform about bugs. This notification should be done by means of an issue tracker, allowing all the information related to the bug, as well as the actor that identifies it, to be stored.

4.4.2 New requirements proposal

During this activity, new requirements can be proposed for the ontology. Ontology developers, domain experts, or users can propose modifications to improve the published ontology version. This proposal should be done using an issue tracker, so that all the information related to it is stored.

5. Conclusions and future work

This document presents the LOT methodology, the methodological framework for OntoCommons. This methodology considers as input information about Industrial Domain Ontologies and Demonstrators to support the challenges found in the domain ontologies landscape and the requirements of the OntoCommons demonstrators.

The LOT methodology supports the entire ontology development process, from the requirements specification to the maintenance of the ontologies, also providing recommendations from the methodological perspective to put it into practice.

The LOT methodology is based on existing methodologies, such as those presented in Section 0 (e.g., NeOn or METHONTOLOGY) and aligned with agile practices taken from Software Engineering. LOT has evolved during the last decade (since its inception in [2]) and it is intended to continue evolving with feedback from its usage, e.g., from the application in OntoCommons demonstrators.

Future work regarding the ontology development methodology will focus on the alignment between the methodological activities proposed in this report and the tools that will be proposed in the landscape analysis of ontological platform support, also following the tool ecosystem that was presented in the report on Ontology Ecosystem Specification and that is shown in Figure 24.

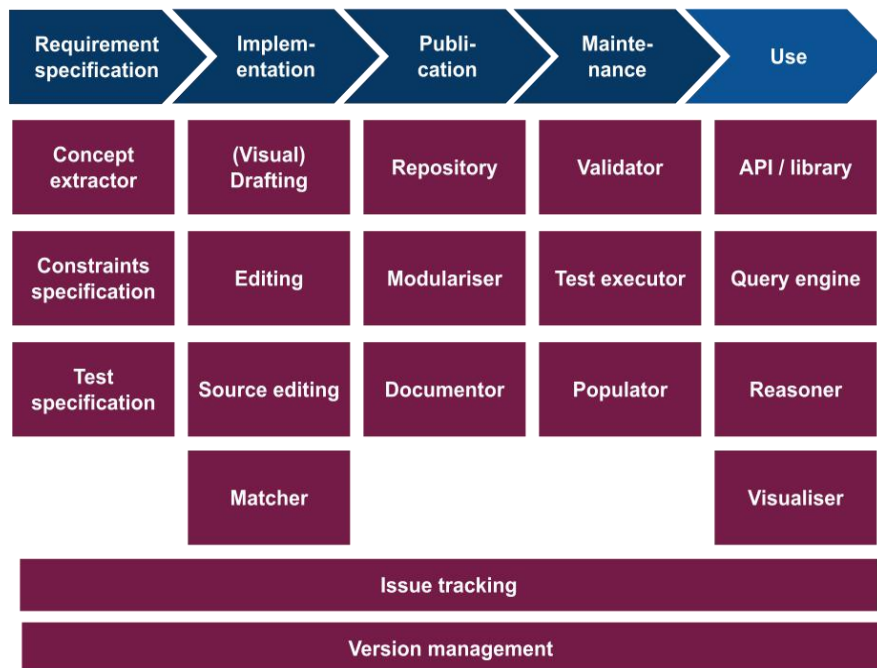


Figure 24 Components of the ontology ecosystem toolkit

Moreover, this methodology will be validated with the participating demonstrators, and the feedback obtained will also be used to improve the methodology. To help such demonstrators in the adoption of the ontology development methodology, a webinar will be organised in September, in which all the activities will be presented, together with recommendations and tips for putting it into practice.

6. References

- [1] M. C. Suárez-Figueroa y A. Gómez-Pérez, «NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology,» de *Proceedings of the International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES*, Sofia, Bulgaria, 2009.
- [2] M. Poveda-Villalón, «A reuse-based lightweight method for developing linked data ontologies and vocabularies,» de *Proceedings of the 9th Extended Semantic Web Conference*, Heraklion, Crete, 2012.
- [3] R. García-Castro, A. Fernández-Izquierdo, C. Heinz, P. Kostelnik, Poveda-Villalón-María y F. Serena, «D2.2. Detailed Specification of the Semantic Model,» 2017.
- [4] S. Chávez, F. Bosché, N. Bountouni, S. Fenz, R. García-Castro, G. Giannakis, S. González-Gerpe, S. Kollmer, S. Kousouris, D. Ntalaperas, T. Thanos, F. Lampathaki, M. Poveda-Villalón, E. Valero, D. Vergeti y F. Tavakolizadeh, «D4.2 BIMERR Ontology & Data Model,» 2020.
- [5] A. Fernández-Izquierdo, A. Cimmino, R. García-Castro, M. Poveda-Villalón, S. Terzi y C. Patsonakis, «T1.3 DELTA Ontology and Data Modelling Framework,» 2019.
- [6] M. Grüninger y M. S. Fox, «Methodology for the Design and Evaluation of Ontologie,» 1995.
- [7] M. G.-P. A. & J. N. Fernández-López, «Methontology: from ontological art towards ontological engineering,» 1997.
- [8] Y. Sure, S. Staab y R. Studer, «On-To-Knowledge Methodology,» de *Handbook on Ontologies*, Springer Berlin Heidelberg, 2004.
- [9] H. S. Pinto, C. Tempich y S. Staab, «Ontology engineering and evolution in a distributed world using DILIGENT,» de *Handbook on ontologies*, 2009.
- [10] S. Auer, «The RapidOWL Methodology--Towards Agile Knowledge Engineering,» de *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2006.
- [11] S. Peroni, «A Simplified Agile Methodology for Ontology Development,» de *OWL: Experiences and Directions – Reasoner Evaluation*, Bologna, Italy, 2017.
- [12] A. Abdelghany, N. R. Darwish y H. A. Hefni, «An agile methodology for ontology development,» *International Journal of Intelligent Engineering and Systems*, vol. 12, nº 2, pp. 170-181, 2019.

- [13] M. C. Suárez-Figueroa, A. Gómez-Pérez y M. Fernández-López, «The NeOn Methodology for Ontology Engineering,» de *Ontology engineering in a networked world*, Springer, Berlin, Heidelberg, 2011.
- [14] M. Poveda-Villalón, P. Calleja-Ibañez, J. Bosque-Gil, E. Montiel-Ponsoda, K. Dimitropoulos y K. Stefanidis, «D3.2 Enriched multilingual ontology with signs in different,» 2018.
- [15] M. Grüninger y M. S. Fox, «Methodology for the Design and Evaluation of Ontologies,» 1995.
- [16] M. Fernández-López, A. Gómez-Pérez y N. Juristo, «METHONTOLOGY: From Ontological Art Towards Ontological Engineering,» de *AAAI-97 Spring Symposium Series*, 1997.
- [17] M. C. Suárez-Figueroa, A. Gómez-Pérez y B. Villazón-Terrazas, «How to write and use the ontology requirements specification document,» de *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, 2009.
- [18] P. Haase, S. Brockmans, R. Palma, J. Euzenat y M. d'Aquin, «D1.1.2 Updated Version of the Networked Ontology Model,» 2006.
- [19] D. Garijo y M. Poveda-Villalón, «Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web,» de *Applications and Practices in Ontology Design, Extraction, and Reasoning*, IOS Press, 2020.
- [20] V. A. Carriero, M. Daquino, A. Gangemi, A. G. Nuzzolese, S. Peroni, V. Presutti y F. Tomasi, «he Landscape of Ontology Reuse Approaches,» de *Applications and Practices in Ontology Design, Extraction, and Reasoning*, 2020.
- [21] M. Fernández-Lopez, M. C. Suárez-Figueroa y Gómez-Pérez-Asunción, «Ontology Development by Reuse,» de *Ontology Engineering in a Networked World*, 2012.
- [22] M. Poveda-Villalón, A. Gómez-Pérez y M. C. Suárez-Figueroa, «OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation,» *International Journal on Semantic Web and Information System*, 2014.
- [23] M. C. Suarez-Figueroa, S. Brockmans, A. Gangemi, A. Gomez-Perez, J. Lehmann, H. Lewen, V. Presutti y M. Sabou, «NeOn Modelling Components. Deliverable 5.1.1 NeOn Project,» 2007.
- [24] Y. S. S. & S. R. Sure, «On-to-knowledge methodology (OTKM).,» de *Handbook on ontologies*, Springer Berlin Heidelberg., 2004, pp. pp. 117-132.
- [25] M. Hristozova y L. Sterling, «An eXtreme method for developing lightweight ontologies,» 2002.
- [26] V. Presutti, E. Daga, A. Gangemi y E. Blomqvist, «eXtreme Design with Content Ontology Design,» de *Proceedings of the 2009 International Conference on Ontology Patterns*, Washington DC, 2009.
- [27] M. Uschold y M. Gruninger, «Ontologies: Principles, methods and applications.,» *The Knowledge Engineering Review*, vol. 11, nº 2, 1996.

- [28] A. Fernández-Izquierdo y R. García-Castro, «Themis: a tool for validating ontologies through requirements,» de *International Conference on Software Engineering and Knowledge Engineering*, Lisbon, Portugal, 2019.
- [29] A. Lawrynowicz y C. M. Keet, «The TDDonto tool for test-driven development of DL knowledge bases,» de *Description Logics*, 2016.
- [30] D. Schober, I. Tudose, V. Svatek y M. Boeker, «OntoCheck: verifying ontology naming conventions and metadata completeness in Protégé 4,» *Journal of Biomedical Semantics*, vol. 3, nº 2, 2012.
- [31] F. Radulovic, M. Poveda-Villalón, D. Vila-Suero, V. Rodríguez-Doncel, R. García-Castro y A. Gómez-Pérez, «uidelines for Linked Data generation and publication: An example in building energy consumption,» *Automation in Construction*, vol. 57, 2015.
- [32] S. Peroni, D. Shotton y F. Vitali, «The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation,» de *Internation conference on Knowledge Engineering and Knowledge Management*, Galway , 2012.
- [33] D. Garijo, «WIDOCO: A Wizard for Documenting Ontologies,» de *International Semantic Web Conference*, Vienna, 2017.
- [34] A. Alobaid, D. Garijo, M. Poveda-Villalón, I. Santana-Perez, A. Fernández-Izquierdo y O. Corcho, «Automating Ontology Engineering Support Activities with OnToology,» *Journal of Web Semantics*, vol. 57, 2019.
- [35] L. Halilaj, N. Petersen, I. Grangel-González, C. Lange, S. Auer, G. Coskun y S. Lohmann, «VoCol: An Integrated Environment to Support Version-Controlled Vocabulary Development,» de *International conference on Knowledge Engineering and Knowledge Management*, Bologna, 2016.
- [36] F. Priyatna, O. Corcho y J. Sequeda, «Formalisation and Experiences of R2RML-Based SPARQL to SQL Query Translation Using Morph,» de *International conference on World Wide Web* , Seoul , 2014.
- [37] A. Cimmino, A. Fernández-Izquierdo y R. García-Castro, «Astrea: Automatic Generation of SHACL Shapes from Ontologies,» de *Extended Semantic Web Conference*, 2020.
- [38] D. Vrandečić, S. Pinto, C. Tempich y Y. Sure, «The DILIGENT knowledge processes,» *Journal of Knowledge Management*, 2005.
- [39] M. Poveda-Villalón, A. Fernández-Izquierdo y R. García-Castro, «Linked Open Terms (LOT) Methodology,» January 2019. [En línea]. Available: <http://doi.org/10.5281/zenodo.2539305>.

Annex 1

This table shows the information collected in the long survey.

Table 4 Information collected in the ontology long survey

Question	Explanation	Possible values
Name	The name was given to the ontology.	Free text
URI	The URI of the ontology.	Free text
Description	A free-text account of the ontology.	Free text
Domains	The different domains covered by the ontology. If the ontology covers more than one domain, please separate them by commas. Example: manufacturing, material science, maintenance, AEC industry, marketing ...	Free text
Scope	The scope of the ontology in a particular domain, e.g., predictive maintenance, stakeholder description, product nomenclature, sensor or building	Free text
Namespace	The preferred namespace URI to use when using terms from this vocabulary.	Free text
Version	The version of the ontology.	Free text
Creation date	The date of formal issuance of the ontology.	Date
Last update	Most recent dates on which the ontology was changed, updated, or modified.	Date
Contact person	The person(s) primarily responsible for making the ontology. Please include the name and email address of the contact persons whenever possible. If there is more than one contact person, please separate them by commas.	Free text
Publisher	The organization that published the ontology.	Free text
Ontology language	The ontology language in which the ontology is implemented.	OWL, RDF-S, SKOS, SUO-KIF, Isabelle, (FOL), OBO format, UML, OntoUML, Other
Format	Format in which the ontology code is provided.	RDF/XML, Turtle, N3, N-Triples, TriX, TriG, Other
Use of top-level ontologies?	Top level ontologies used by the ontology.	Basic Formal Ontology, DOLCE, SUMO, EMMO,

		unified Foundational Ontology, YAMATO, CYC, General Formal Ontology, Other
License	The license of the ontology. Example: CC BY-SA, MIT, etc.	All rights reserved / no license (No Open), CC0 1.0 Universal, CC-BY International, CC-BY Unported, CC-BY-SA International, CC-BY-SA Unported, CC-BY-ND, CC-BY-NC, CC-BY-NC-SA, CC-BY-NC-ND, GFDL, MIT, PDDL, ODC-By, ODBL, W3C software license, Unknown, Other
Please specify (license)	Specify the license if it is not one of the list.	
Language	The ISO 639-1 code(s) of the language(s) of the resource. If the ontology is implemented in more than one language, please separate them by commas. Example: es, en, (See http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes for a full list of codes).	en – English, es – Spanish, fr – French, de – German, it – Italian, bg – Bulgarian, nl – Dutch, no – Norwegian, ru – Russian, Other
Available documentation	URLs for the documentation of the ontology (for example a website)	Free text
References	Resources that might provide additional information (documents, deliverables, papers, etc.).	Free text
Ontology registered	Is the ontology stored and indexed in a dedicated repository/registry? If yes, could you please specify which one and provide the URL of the repository/registry?	Free text
Best practices	Free text	OBO Foundry, Industry Ontology Foundry principles, FAIR Principles, None, Other

Development methodology and knowledge sources	Please provide a short description of the methodology and knowledge sources used to develop the ontology as a comma separated list	Free text
Is the ontology an outcome of a European project? If so, please indicate the project name and the website if possible.	Whether the ontology has been developed in one or more European projects.	Free text
Is the ontology developed within a standardization body? If yes, please specify which one	Whether the ontology has been developed in the context of standardization bodies.	Free text
Is the ontology based on any standard? If yes, please specify which one(s)	Whether the ontology is based on existing standards.	Free text
Is the ontology supported by a community? If yes, please mention the involved community(ies)	Whether the ontology is being supported by any community.	Free text
Is there a sustainability plan for this ontology?	Whether there is a sustainability plan form an organization, community, company, etc.	None, Yes, No, Maybe, Unknown
Is the ontology being reused by other ontologies or projects? If yes, could please specify which ones?	Whether the ontology is being adopted.	Free text
Is the ontology aligned with other ontologies, reuse other ontologies or specific design patterns? If yes, please specify which one(s).	Whether the ontology reuses ontology design patterns.	Free text
Comments	Further information about the ontology that might be relevant.	Free text