Engineering Applications of Artificial Intelligence 111 (2022) 104755

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

LOT: An industrial oriented ontology engineering framework

María Poveda-Villalón [a,*], Alba Fernández-Izquierdo [a], Mariano Fernández-López [b], Raúl García-Castro [a]

[a] Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Madrid, Spain
[b] Escuela Politécnica Superior, Universidad San Pablo CEU, Madrid, Spain

ARTICLE INFO

Keywords:
Ontology engineering
Ontology development methodology
Ontology development software support
Collaborative ontology development
Ontology industrial development

ABSTRACT

Ontology Engineering has captured much attention during the last decades leading to the proliferation of numerous works regarding methodologies, guidelines, tools, resources, etc. including topics which are still being investigated. Even though, there are still many open questions when addressing a new ontology development project, regarding how to manage the overall project and articulate transitions between activities or which tasks and tools are recommended for each step. In this work we propose the Linked Open Terms (LOT) methodology, an overall and lightweight methodology for building ontologies based on existing methodologies and oriented to semantic web developments and technologies. The LOT methodology focuses on the alignment with industrial development, in addition to academic and research projects, and software development, that is making ontology development part of the software industry. This methodology includes lessons learnt from more than 20 years in ontological engineering and its application on 18 projects is reported.

1. Introduction

Q1[1]: Should you always write ontology functional requirements in the form of Competency Questions (CQs) (Grüninger and Fox, 1995) when you elaborate the requirements specification for an ontology? Q2: Which is the best way to communicate requirements to software engineers? Q3: And to domain experts? Q4: Which are the main problems that you will find when you reuse an ontology? Q5: Which is the sequence of activities that has been shown to be successful in practice in ontology development? Q6: What tools can you use for each activity in ontology development? Q7: Are there some tools that aid you to identify typical mistakes in modeling? Q8: What do you have to do to transform your Web Ontology Language (OWL) ontology into an HTML document?

Although it is true that a lot of useful work has been carried out on ontology engineering over the years, such as the proposal of multiple ontology development methodologies to systematize the development process and the alignment with agile practices, as of today, there are important questions on ontology development that have not been answered, This issue has been exposed by the recent analysis of the current state, challenges and future directions in ontology engineering presented by Tudorache (Tudorache, 2020).

The aim of the work presented in this paper is to respond to this situation and answer the questions presented in the first paragraph by proposing the Linked Open Terms (LOT) methodology, which not only presents the activities to be performed in the ontology development process, but also proposes recommendations, tips and tools to support them. The LOT methodology is based on the experience of, at least, 18 projects where ontologies have been developed, both by this paper authors and by external teams, involving both domain experts and software engineers. Our experience is also diverse in other senses, for example, there are projects where the creation of linked open data has been an important result, others where the ontology has been an objective itself, others where the ontology has been an standard schema for communication between systems, etc. In addition, one of the authors has contributed in the past to two of the most well-known methodologies for building ontologies which brings not only an extensive experience in practical matters but also a broader view and knowledge about the evolution of the ontology engineering field during the last decades. The conclusions and lessons learnt from our experience are presented here as a framework that includes both the methodological and the technological level.

The paper is structured as follows. Section 2 presents main existing methodologies for developing ontologies. Section 3 describes the methodological level of the LOT framework which includes the proposed activities to be performed in any ontology development process, while Section 4 presents the software support recommended to carry out such activities. Section 5 focuses on the validation of the

* Corresponding author.
E-mail addresses: m.poveda@upm.es (M. Poveda-Villalón), albafernandez@fi.upm.es (A. Fernández-Izquierdo), mfernandez.eps@ceu.es (M. Fernández-López), r.garcia@upm.es (R. García-Castro).

[1] The following questions are identified by "QX" to ease the reference to the questions from other sections of the paper.

https://doi.org/10.1016/j.engappai.2022.104755
Received 2 March 2021; Received in revised form 24 January 2022; Accepted 5 February 2022
Available online 3 March 2022

methodology based on its application and comparison with existing approaches. Finally, Section 6 concludes and presents future lines of work.

2. Related work

Although numerous ontology development methodologies have been proposed since the 1990s, existing methodologies should be reviewed and adapted to support ontology development in the Linked Data and agile context, which are currently popular scenarios. This section provides an overview of ontology development methodologies mainly oriented to OWL ontologies.

Traditional well-known methodologies, such as METHONTOLOGY (Fernández-López et al., 1997), On-To-Knowledge (Staab et al., 2001), DILIGENT (Pinto et al., 2004) and the "Ontology Development 101" guide (Noy and McGuinness, 2001), were proposed before the Linked Data paradigm arose and when agile methodologies for software and ontology development did not have the impact that they have nowadays. Therefore, there are gaps of aspects not considered by them (e.g., ontology publication and reuse based on web mechanisms).

Specifically, METHONTOLOGY defined a set of life cycle models and a development process that provided an overview of how an ontology should be developed. Moreover, it provided detailed guidelines to carry out the ontology conceptualization. The life cycle models that it proposed were waterfall, incremental (which ensures that each version is compatible with the previous ones) and one based on evolving prototypes (with essential similarities to agile development). However, from a current perspective it has some drawbacks: (1) there are activities that are not defined in a precise way; (2) it was focused on developing application-independent ontologies, however, currently it is usual to develop ontologies as part of a bigger software project; (3) some of the premises associated with METHONTOLOGY are no longer valid, for instance, that the ontologies that you reuse will be stable and available forever; and (4) there was not so much experience as now developing ontologies.

On-To-Knowledge (Staab et al., 2001) was a valuable contribution to show ontology development as a part of knowledge-based systems. Nevertheless, it did not provide as many details as METHONTOLOGY for ontology conceptualization and shared most of its drawbacks.

With regards to the "Ontology Development 101" guide (Noy and McGuinness, 2001), it is specially useful for ontology conceptualization. However, it has the following limitations: (1) the requirements specification is just based on competency questions, but no other complementary or alternative techniques, which have been shown to be valuable in practice, are taken into account; (2) the reuse activity is only described in one paragraph, although it is an issue with a lot of shades (Fernández-López et al., 2019); and (3) it has the issues mentioned for METHONTOLOGY.

Concerning DILIGENT (Pinto et al., 2004, 2009), it is focussed on collaborative development and, therefore, it cannot be considered a proposal for the whole development process of an ontology.

NeOn (Suárez-Figueroa et al., 2015), another well-known methodology, identifies nine flexible scenarios for building ontology networks, including scenarios related to the reuse of ontological resources, the reuse and reengineering of non-ontological resources or the reuse of ontology design patterns (ODPs) (Gangemi and Presutti, 2009). NeOn (Suárez-Figueroa et al., 2015) was a combined European effort[2] to provide precise guidelines, supported by an integrated development environment (the NeOn Toolkit[3]), to develop ontology networks. An *ontology network* is a collection of ontologies related together via meta-relations such as import, versioning, etc. Suárez-Figueroa et al. (2012). It took into account new techniques not considered by either METHONTOLOGY or OnToKnowledge as, for example, the reuse of ontology design patterns (ODPs) or of non-ontological resources (e.g. spreadsheets). Nevertheless, Neon has the following drawbacks: (1) it establishes the generation of a quantity of documentation that provokes that, in practice, this methodology is not followed as it is; (2) its proposals were based on the best knowledge at the time it was published; however, after the experience of several years, are currently applied with important variations; and (3) a community has not been maintained over time to support the NeOn Toolkit and, therefore, it has been discontinued.

All these traditional methodologies, although they include support activities, such as evaluation, propose time and resource consuming activities to develop and evaluate ontologies instead of simple and (semi-)automatic processes, being sometimes too heavy to be applied in ontology developments. However, these methodologies are reused and proposed to be followed as part of LOT when the techniques are still applicable.

More recent approaches introduce agile methodologies for ontology development, focused on flexible and iterative development of ontologies that allows to have in short periods of time several versions of ontologies which, although they are not complete, can be consumed by users or domain experts. On the one hand, Hristozova and Sterling propose the eXtreme Method (Hristozova and Sterling, 2002), inspired by Extreme programming (Beck, 2000). However, we are not aware of whether this idea has been validated through the execution of different projects. Therefore, the applicability conditions of this approach are not clear. Moreover, their work is shown in a short paper and, consequently, although its principles can be useful for further work, the proposal cannot be taken as a detailed guide to develop ontologies. Regarding the XD methodology (Presutti et al., 2009), it is a collaborative, incremental and iterative method for pattern-based ontology design; therefore, other aspects (e.g. documentation, publication, etc.) are out of the scope of the proposal and it only consider the case for reusing ontology patterns rather than other resources.

On the other hand, SAMOD (Peroni, 2016) is the result of its author's dedication to the development of ontologies for six years, and it has been evaluated through a controlled experiment. Nevertheless, it is focused on the sequence of activities for ontology development and the criteria for transition from one state of the development to another. It is true that some guidelines and examples are provided to elaborate test cases (the SAMOD corner stone); however, precise recommendations for conceptualization, documentation, tools to be used, etc. are out of the scope of SAMOD. Moreover, no alternatives according to different situations are not presented to the proposed techniques. Finally, we have no record of updating of this methodology after 2016. RapidOWL (Auer, 2006) is a set of values, principles and practices based on agile development, but no precise guidelines to develop ontologies are provided. Moreover, we are not aware of this proposal has been refined in projects after 2006.

There are also lightweight methodologies,[4] such as UPON Lite (De Nicola and Missikoff, 2016), which intent to place end users at the center of the process. UPON Lite (De Nicola and Missikoff, 2016) starts from the premise that ontologies can be developed by domain experts with a minimal intervention of ontology engineers. However, so far, no one has managed to prove this fact or, at least, under which conditions this premise is true. Furthermore, it shares some of the limitations shown for the aforementioned methodologies, in particular: (1) we have no record of updates after 2016, and (2) it provides guidelines for conceptualization, but other activities are out of the scope of the proposal.

All these aforementioned methodologies describe the set of activities that should be carried out in any ontology development process, which includes encoding of the ontology and only in some cases also

[2] http://neon-project.org/nw/Welcome_to_the_NeOn_Project.html.

[3] http://neon-toolkit.org/wiki/Main_Page.html.

[4] Lightweight methodologies in this context refer to methodologies oriented to rapid ontology engineering or that are based on few rules generally easy to follow.

the support activities and none of them consider the publication activity and related tasks. Therefore, LOT is the first methodology oriented to the publication of ontologies following semantic web best practices and FAIR principles including specific recommendations, tips and potential tools that can be helpful to ontology developers.

This paper is focused on both methodological and technological aspects that are needed during any lightweight ontology development process. It aims to go beyond previous methodologies by helping ontology developers to build ontologies taking into account their particular characteristics, following a lightweight approach and providing detailed methodological guidelines for the proposed activities.

3. Methodology

The Linked Open Terms (LOT) methodology is a lightweight methodology for developing ontologies and vocabularies focusing on industry projects. This section provides details about LOT,[5] the methodology which was first proposed in Poveda-Villalón (2012) and further developed in García-Castro et al. (2017). This methodology aims to be compatible with software development techniques in which sprints and iterations represent the main workflow organization in order to align the ontology development with software development agile practices. In addition, the methodology focuses on: (a) the reuse on terms (ontology classes, properties and attributes) existing in already published vocabularies or ontologies and (b) on the publication of the built ontology according to Linked Data principles. It is also worth mentioning that the LOT methodology builds on top of the ontological engineering activities defined in the NeOn methodology (Suárez-Figueroa et al., 2015) when available.

The LOT methodology defines iterations over a basic workflow composed of the following activities (see Q5 in Section 1): (1) Ontological requirements specification; (2) Ontology implementation; (3) Ontology publication; and (4) Ontology maintenance. The activities, roles involved and main expected outputs are depicted in Fig. 1. The main workflow of LOT is inspired by core workflows of existing methodologies where the sequence "Requirements elicitation-Implementation-Evaluation" appears. This main workflow has been enriched with Semantic Web oriented best practices and goals as the ontology publication, activity not taken into account in previous methodologies. The LOT methodology also combines and offers different ways of describing ontology requirements (Competency Questions (Grüninger and Fox, 1995), Natural language statements and tabular information based on METHONTOLOGY) instead o a fix or unique technique. It also provides a template for ontology functional requirements description including information to ease traceability. Another novelty of the LOT methodology is the alignment with software development practices and tools as well as the inclusion of the continuous integration notion in the ontology developments. Finally, the LOT methodology include tips and recommendations based on more than 20 years of experience for each activity.

Even though there are many roles which can be involved in ontology development projects, the LOT methodology classifies them into the following groups:

- **Ontology developer:** An ontology developer is a member of the ontology development team who has high knowledge about ontology development and knowledge representation. Ontology developers could be further specified as an ontology engineer; however, for the sake of simplicity, only the ontology developer term will be used in the paper.
- **Domain expert:** A domain expert is an expert in the domains covered by the ontology. This role does not need knowledge about ontology development.

[5] Online documentation of the methodology is available at https://lot.linkeddata.es/.

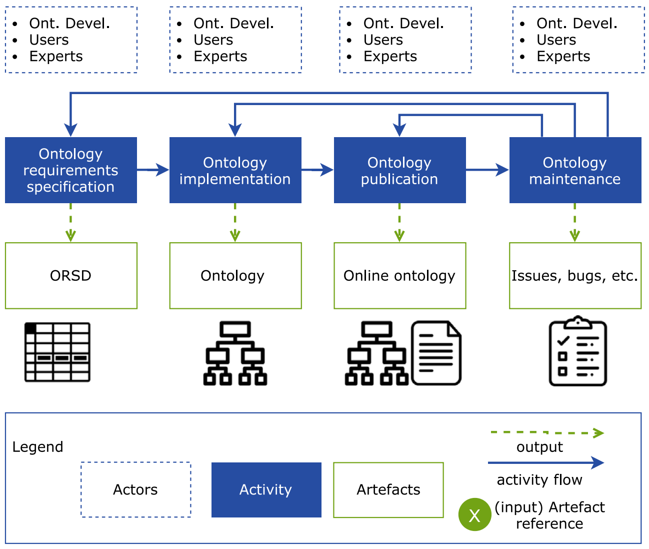


Fig. 1. LOT methodology base workflow.

- **Ontology user:** An ontology user is a potential end user of the ontology. This actor also includes software developers who will make use of the ontology within their applications. Even though there might be multiple types of users, such as the users interacting with the applications built on top of the ontologies, in this work we refer as "user" actors who would potentially use the ontology directly rather than embedded in broader systems.

Details including more fine grained activities and workflows, intermediate results and tools supporting the activity are provided in the following sections for each activity.

3.1. Ontology requirements specification

The ontology requirements specification activity refers to the activity of collecting the requirements that the ontology should fulfill (Suárez-Figueroa et al., 2015). These requirements are usually related to the goal of the ontology, to the domain that the ontology should model, or to technical details of the ontology like the implementation language, among others. In this step, involvement and commitment by experts in the specific domain at hand is required to generate the appropriate industry perspective and knowledge.

The workflow proposed for the ontology requirements specification activity is shown in Fig. 2. In this and following figure, one of the main activities are decomposed in sub activities (yellow filled boxes within the activity being detailed) for example, in this figure the "Ontology requirements specification" box from Fig. 1 has been converted in a broader box including its sub-activities. The numbers in ellipses are used in two situations: a) when they appear at the bottom of a white box they are used to identify the resource being produced as output of a sub-activity, for example the first appearance of the "Functional Ont requirements (verified)" is identified by the number 5; and b) when the number appears at the top of a filled box it represents the resources taken as input, for example the number 5 appears attached to the sub-activity "ORSD formalization" meaning that the "Functional Ont requirements (verified)" are taken as input.

For the case of the ontology requirements specification activity, it is planned to start with two complementary sub-activities, namely "Use case specification" and "Data exchange identification", from which at least one should be carried out in order to start gathering documentation to help the knowledge elicitation activity (Fernández-López et al., 1997). Depending on the project configuration, for example, whether the ontology is defined to model a domain in a general level fashion or whether specific data from an API or database is intended to be transformed, the ontology development team might carry out one or two of the sub-activities.

This activity 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.

3.1.1. Use case specification

The goal of the **use case specification** activity is to provide a vision on 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. The use cases, in the context of ontology development, allow describing situations that are desired to be reached with the data that are described by the vocabulary. Therefore, their objective is to guide the obtaining of the requirements. The specification of a use case consists of the narration of a scenario in which one or several actors intervene and the necessary interactions to obtain an expected result or benefit. Below an example of a use case is presented in which a vision is given about how the data could be used. This activity is recommended to be carried out in any ontology development project and highly advised for those in which the ontology aims to be defined in a top level or general way.

An example of a use case extracted from the *Ciudades Abiertas* ontologies development project (Espinoza-Arias and Corcho, 2018) is shown below:

> *Name*: Use Case 1 - Real Time Parking
>
> *Description*: Finding available parking spaces 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 in the streets and has made available to the public a mobile application for guided parking. Thanks to this system, the citizens can know in real time in which public car park there are free places to leave their car.
>
> *Actors*: citizen, application
>
> *Flow*: Mike, while driving his car to the city center, activates through a voice command the parking application. 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, however the spot is not reserved for him. Mike follows the route suggested by the application, arrives at the place and parks.

3.1.2. 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 modeled. In this case, the documentation to be shared might correspond to: datasets, regulations, standards, data formats, APIs specifications, database schemas, etc. Typically in this activity, ontology users such as software developers are responsible for providing this documentation to ontology engineers. The output of this activity is a set of domain documents and resources. An example of this document exchange is shown in Fig. 3(a) and Fig. 3(b) in which the Thing Description specification from the W3C Web of Things Working Group[6] and a data sample of the same specification are depicted, respectively. This data exchange example is taken from the development of the VICINITY ontology network (García-Castro et al., 2017).

3.1.3. Purpose and scope identification

From use cases and the domain documentation provided in the data exchange identification task, the **purpose and scope identification** activity is carried out. The ontology development team works in collaboration with users and domain experts to define the purpose and scope of each ontology or ontology module to be developed. The communication between the domain experts, users and ontology development team could be carried out by means of online or physical meetings. As output of this activity a text document describing the purpose and scope of the ontology is produced. This document could include non-functional 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.

3.1.4. Functional ontological requirements proposal

Subsequently, the task of elaborating the **functional ontology requirements proposal** is carried out mostly by the ontology engineers, but it could be supported by users and domain experts. This activity is expected to produce a set of functional requirements that could be materialized into one or more of the following forms:

- **Competency questions (CQs):** This is one of the well-known techniques to define ontology functional requirements. This technique was proposed by Grüninger and Fox (1995) and suggests to elaborate a set of queries that the ontology to be developed should answer. This set of queries is considered as requirements of the ontology, more precisely, informal competency questions, since they are not yet expressed in a formal language. The competency questions might be accompanied by answers or expected results.
- **Natural language statements:** In this case, the domain to be represented in the ontology is described by writing affirmative or negative natural language statements. This technique is an alternative to CQs that could be used in combination with competency questions. This technique is similar to the software specification technique, where each requirement unambiguously defines a need that must be included in the software product (IEEE, 1998).
- **Tabular information:** This technique was proposed in METHONTOLOGY and consists of creating 3 types of tables:
  - **Concepts**: this table is used for collecting information about the concepts, alternative identifiers and a short description for each of them to clarify their meaning within the domain.
  - **Relations**: this table is used to group all the relations between the concepts. Also, additional information is provided such as a description, maximum cardinality, whether they are symmetric, transitive, functional, etc.
  - **Attributes**: this table is used to group all related attributes under the relevant concepts of the terms table, i.e., relations to be established between concept instantiations and particular data values.

While traditionally competency questions have attracted attention from many methodologies, LOT proposes to use also other techniques depending on the case at hand or to combine them (see Q1 in Section 1). In general, the competency questions and natural language statements are commonly used in combination. To get some examples, the CORAL Corpus (Fernández-Izquierdo et al., 2019b) can be used. CORAL[7] includes two components: (1) a dictionary of lexico-syntactic and (2) 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

[6] https://www.w3.org/WoT/WG/.

[7] http://coralcorpus.linkeddata.es/.

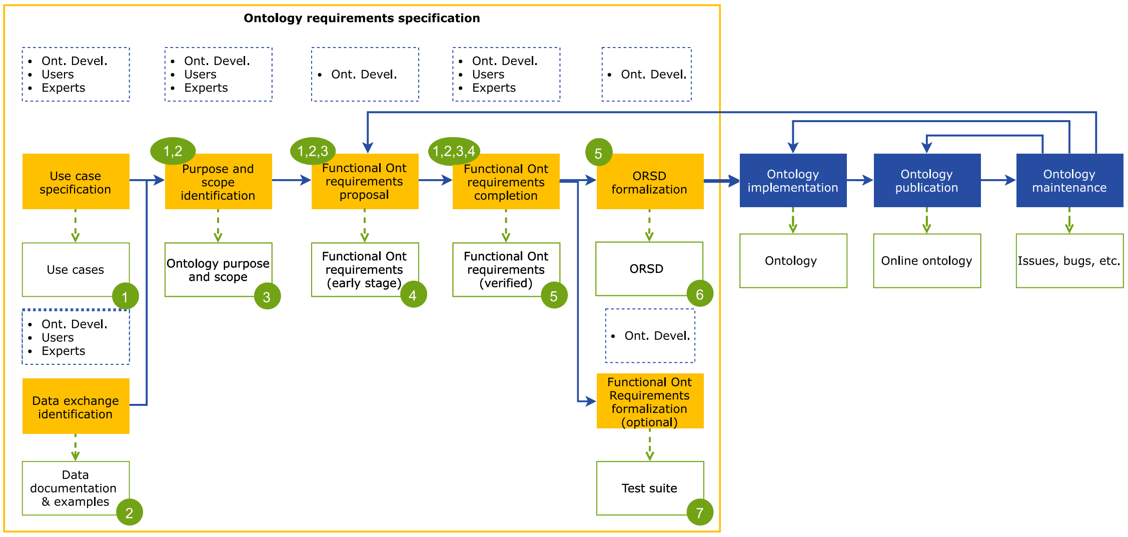
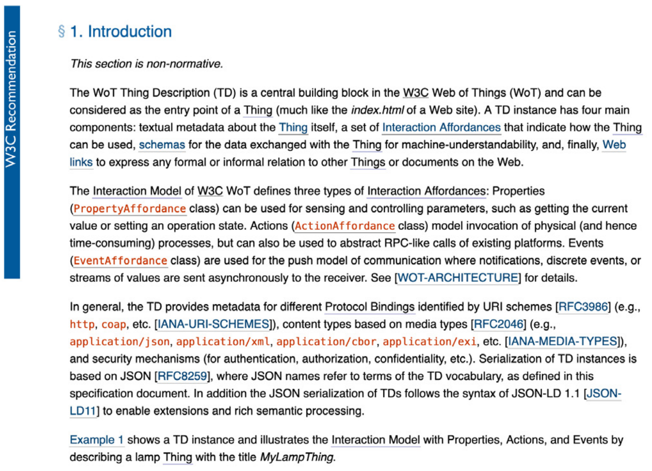

Ontology requirements specification
Ont. Devel.
Users
Experts
Use case specification
Use cases
Data exchange identification
Data documentation & examples
Purpose and scope identification
Ontology purpose and scope
Functional Ont requirements proposal
Functional Ont requirements (early stage)
Functional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

Fig. 2. Workflow proposed for ontology requirement specification.


W3C Recommendation

§ 1. Introduction

This section is non-normative.

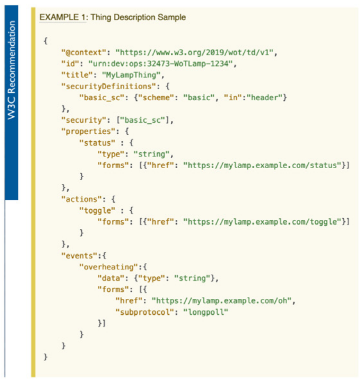
The WoT Thing Description (TD) is a central building block in the W3C Web of Things (WoT) and can be considered as the entry point of a Thing (much like the *index.html* of a Web site). A TD instance has four main components: textual metadata about the Thing itself, a set of Interaction Affordances that indicate how the Thing can be used, schemas for the data exchanged with the Thing for machine-understandability, and, finally, Web links to express any formal or informal relation to other Things or documents on the Web.

The Interaction Model of W3C WoT defines three types of Interaction Affordances: Properties (`PropertyAffordance` class) can be used for sensing and controlling parameters, such as getting the current value or setting an operation state. Actions (`ActionAffordance` class) model invocation of physical (and hence time-consuming) processes, but can also be used to abstract RPC-like calls of existing platforms. Events (`EventAffordance` class) are used for the push model of communication where notifications, discrete events, or streams of values are sent asynchronously to the receiver. See [WOT-ARCHITECTURE] for details.

In general, the TD provides metadata for different Protocol Bindings identified by URI schemes [RFC3986] (e.g., `http`, `coap`, etc. [IANA-URI-SCHEMES]), content types based on media types [RFC2046] (e.g., `application/json`, `application/xml`, `application/cbor`, `application/exi`, etc. [IANA-MEDIA-TYPES]), and security mechanisms (for authentication, authorization, confidentiality, etc.). Serialization of TD instances is based on JSON [RFC8259], where JSON names refer to terms of the TD vocabulary, as defined in this specification document. In addition the JSON serialization of TDs follows the syntax of JSON-LD 1.1 [JSON-LD11] to enable extensions and rich semantic processing.

Example 1 shows a TD instance and illustrates the Interaction Model with Properties, Actions, and Events by describing a lamp Thing with the title *MyLampThing*.

(a) Web of Things specification excerpt


W3C Recommendation

EXAMPLE 1: Thing Description Sample

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic", "in":"header"}
    },
    "security": ["basic_sc"],
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{"href": "https://mylamp.example.com/status"}]
        }
    },
    "actions": {
        "toggle" : {
            "forms": [{"href": "https://mylamp.example.com/toggle"}]
        }
    },
    "events":{
        "overheating":{
            "data": {"type": "string"},
            "forms": [{
                "href": "https://mylamp.example.com/oh",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

(b) Web of Things data sample

Fig. 3. Data exchange identification example extracted from Web of Things documentation (Käbisch et al., 2020) containing both textual description and data samples.

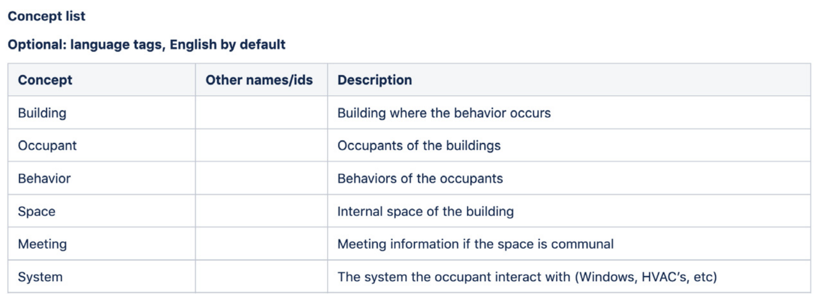
ontology named NP<class>. This dictionary of patterns presents a set of ambiguous expressions that can lead to multiple implementations in the ontology. As an example, the use of the verb "to have" in a requirement can be translated both as a datatype property and an object property in the ontology. These ambiguous expressions should be avoided from the requirements specification since they hinder the formalization of requirements and, therefore, the translation from the requirements to the ontology. Moreover, the set of 834 annotated requirements can be taken as reference or inspiration for ontology requirements definition.

Figs. 4–6 show examples of METHONTOLOGY tables inside a wiki for extracting requirements during the BIMERR European project[8] in which several ontologies about domains involved in building renovation processes for energy efficiency were developed, for example: buildings, weather, sensors and actuators, processes, notification, etc.

If the set of requirements are written in the form of competency questions or natural language sentences, it can also be stored following a tabular approach and include additional information, such as the following fields:

- Requirement identifier, which needs to be unique for each requirement to be used during the whole ontology development process.
- The domain of the requirement identified (in case there are several domains involved in the ontology or ontology network).
- The competency question or natural language sentence.
- The answer to the competency question (in case the previous field is a CQ).
- Status of the requirement, which can be: (1) Proposed, (2) Accepted, (3) Rejected, (4) Pending, (5) Deprecated or (6) Superseded.
- In case the requirement is superseded, the identifier of the updated requirement.
- Comments about the requirement.
- Provenance of the requirement (e.g., if it is reused, standard or document where it comes from, meeting minutes when the requirements was originated).
- Priority of the requirement, which can be: (1) High, (2) Medium or (3) Low.

[8] https://bimerr.eu/.

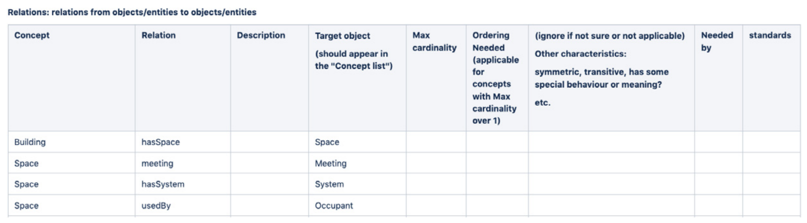


Concept list

Optional: language tags, English by default

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)

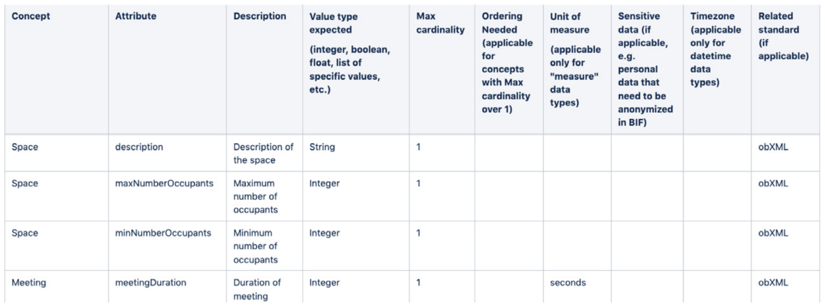
Fig. 4. "Concepts" table from the BIMERR ontology requirements for the occupancy behavior ontology.



Relations: relations from objects/entities to objects/entities

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					

Fig. 5. "Relations" table from the BIMERR ontology requirements for the occupancy behavior ontology.



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

Fig. 6. "Attributes" table from the BIMERR ontology requirements for the occupancy behavior ontology.

Tip: Functional ontology requirements specification

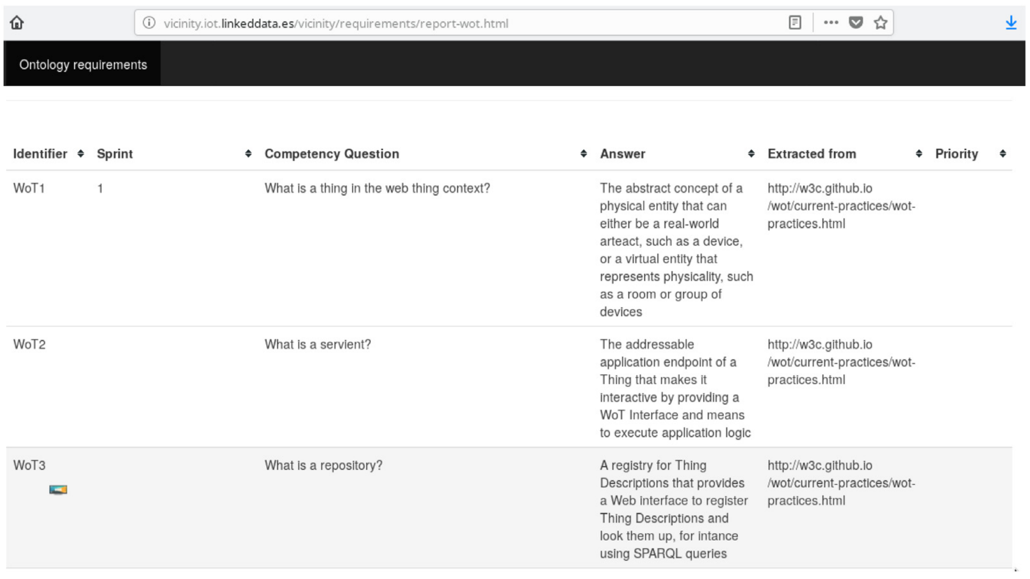
From our experience, it has been observed that using the competency questions technique is more common among ontology engineers or teams with experience in query languages, mostly SPARQL (see Q2 and Q3 in Section 1). However, since domain experts are sometimes not familiar with the OWL and SPARQL capabilities, they could tend to generate more general questions not so much related on how the data is going to be queried. In other cases, it might be unnatural or forced to reformulate as a question knowledge descriptions that are usually expressed as statements, for example transforming "A sensor observes a property" to a question like "What is observed by a sensor?". In these cases the natural language statements help. Also, if there is already structured information about the domain, tables are also useful and more practical for generating a first conceptualization, because they serve as an intermediate representation between the domain and the model. In general we could draw two general recommendations about which technique to use to specify functional requirements:

- Avoid competency questions when domain experts have no knowledge about ontology data generation and querying.
- Use tables if there are structured data or APIs, or there is a close collaboration with software developers.

3.1.5. Functional ontological requirements completion

After the requirements proposal, the **functional requirements completion** task is performed. 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. The following criteria, which does not aim to be an exhaustive list, can be used in this validation task as stated in Grüninger and Fox (1995):

- A set of requirements is correct if each requirement refers to some features of the ontology to be developed.
- A set of requirements is complete if users and domain experts review the requirements and confirm that they are not aware of additional requirements.
- A set of requirements is internally consistent if no conflicts exist between them.
- A set of requirements is verifiable if there is a finite process with a reasonable cost that tests whether the final ontology satisfies each requirement.
- A set of requirements is comprehensible if each and every requirements is understandable to users and domain experts.
- A set of requirements is unambiguous if each and every requirements has only one possible interpretation; that is, if it does not admit any doubt or misunderstanding.
- A set of requirements is concise if each and every requirement is relevant, and no duplicated or irrelevant requirements exist.


vicinity.iot.linkeddata.es/vicinity/requirements/report-wot.html
Ontology requirements

Identifier	Sprint	Competency Question	Answer	Extracted from	Priority
WoT1	1	What is a thing in the web thing context?	The abstract concept of a physical entity that can either be a real-world arteact, such as a device, or a virtual entity that represents physicality, such as a room or group of devices	http://w3c.github.io /wot/current-practices/wot-practices.html	
WoT2		What is a servient?	The addressable application endpoint of a Thing that makes it interactive by providing a WoT Interface and means to execute application logic	http://w3c.github.io /wot/current-practices/wot-practices.html	
WoT3		What is a repository?	A registry for Thing Descriptions that provides a Web interface to register Thing Descriptions and look them up, for intance using SPARQL queries	http://w3c.github.io /wot/current-practices/wot-practices.html	

Fig. 7. Snapshot of ontology requirement specification document in HTML.

If needed, functional requirements are assigned priorities. The main implication of this prioritization is the possibility of planning and scheduling the development of the ontology in sprints. This prioritization would drive the ontology development process since to carry out this prioritization the ontology development team works with the domain experts to identify which requirements need to be fulfilled first. The communication between the domain experts and the ontology development team could be carried out by means of on-line or in-person interviews.

3.1.6. ORSD formalization

Once the ontology development team has all the information about the requirements, obtained from the "purpose and scope identification", "ontological requirements proposal" and "ontological requirements completion" activities, the Ontology Requirements Specification Document (ORSD) is created. This specification document stores all the functional and non-functional requirements identified and the information associated to them. Fig. 7 shows an excerpt of an ORSD from the VICINITY Web Of Things ontology. Note that in this case it represents the HTML version of the requirements published together with the ontology documentation.

LOT proposes a template for the ORSD (in docx and tex formats), which is stored a GitHub repository to be used by users.[9] Such template is an adaptation of the ORSD proposed in the NeOn methodology. However, not all the fields included in the NeOn methodology are included in this ORSD template. For example, the "user" is not considered a mandatory field as in many situations the ontology is transparent to them and it does not affect the ontology development process. It has been observed that finding potential final users of the ontology in some situations, for example when defining an ontology for a standardization body as common vocabulary instead of as a software component, can lead to a time consuming tasks that does not revert in specific requirements or considerations during the development.

Another optional change is the pre-glossary of terms with their frequencies, which refers to the list of terms included in the CQs and their frequencies. These frequencies are calculated to have a measure of the most important terms appearing in the competency questions or natural language statements. That is, to separate domain-repeated concepts from the rest of words. However, if the requirements are taken in the form of tables, this task and field is not needed as the relevant terms are already listed in the concepts table. If the requirements have been taken in the form of competency questions or natural language statements an alternative for the terms frequency is to generate cloud tags to have a more compact view of the terms frequency.

3.1.7. Functional ontology requirements formalization

Finally, the functional ontology requirements formalization can be performed. 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. Instead of adding SPARQL queries, test expressions following the testing method defined in Fernández-Izquierdo and García-Castro (2018) can also be added and stored in RDF files following the Verification Test Case ontology.[10] Furthermore, this ontology can also be used to store the ontology requirements as RDF files. The SPARQL queries and tests can be executed over the ontology to verify if the ontology satisfies the ontological requirements identified later in the ontology development process, more precisely during the ontology evaluation. The definition and execution of these tests allow the requirements traceability during the ontology development process.

3.2. Ontology implementation

The aim of the ontology implementation activity 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). After defining the first set of requirements, though modification and addition of requirements is allowed during the development, the ontology implementation phase is carried out through a number of sprints. The ontology developers schedule and plan the ontology development according to the prioritization of the requirements in the ontology requirements specification process. 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.

The ontology implementation is usually divided into sub-activities as shown in Fig. 8:

[9] https://github.com/oeg-upm/LOT-resources.

[10] https://w3id.org/def/vtc#.

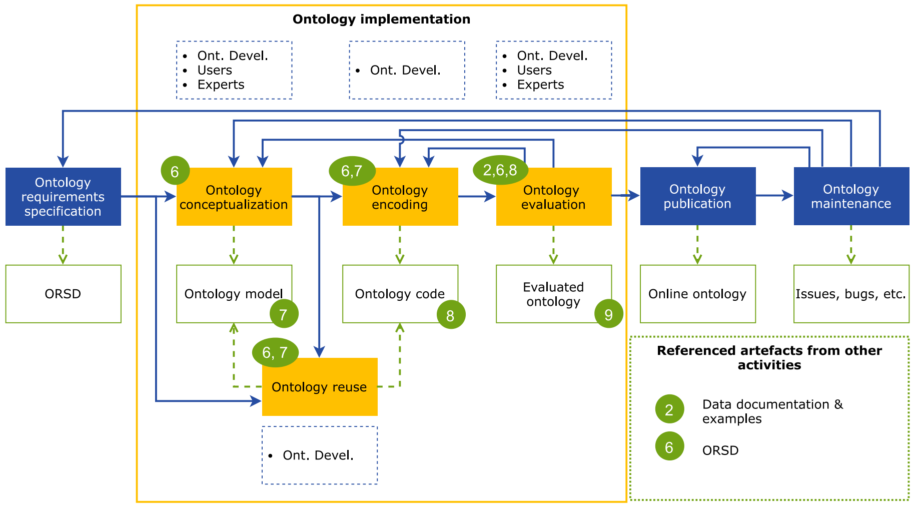


Fig. 8. Workflow proposed for ontology implementation.

3.2.1. *Ontology conceptualization*

The purpose of the conceptualization sub-activity 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.

The conceptualization could be carried out by stating the concepts and relations in a formal system (or logic language) or could be done by means of diagrams, which is currently a common technique. Such diagrams are also useful to document the ontology (see Section 3.3). When using diagrams, either on paper or using a diagramming tool, it is important to decide the notation to be used as it will be part of the common understanding between domain experts and ontology developers and it will also drive the future ontology encoding. The Chowlk notation[11] which is a UML based notation for ontology diagrams; besides the UML_Ont profile (Haase et al., 2009) or the entity relationship diagrams (Chen, 1976) can be used for this purpose. Working in a high level of detail, both domain experts and users can collaborate in the conceptualization of the ontology, helping ontology developers to identify relationships and classes to be added. It is worth to note that, even though they collaborate in the conceptualization, the ontology developers are the ones who are responsible for adding new terms to the ontology.

To develop the conceptualization, a common practice is to incrementally identify the following ontology elements:

1. **Classes** (or concepts): A class is instantiated by individuals that represent entities that have something in common. An individual represents a particular entity of a domain.
2. **Class hierarchies**: A class A is subclass of another class B if and only if every instance of A is also an instance of B.
3. Properties between classes (**relationships**): A binary relationship is instantiated by pairs of individuals that represent facts.
    - A specially important binary relation is *part of* Varzi (2003), which must not be confused with *subclass of*.
    - There might be relations instantiated by tuples with more than two individuals, which are called n-ary relations. In this case we refer to the "Defining N-ary Relations on the Semantic Web" W3C Semantic Web Best Practices and Deployment Working Group Note[12] for more information about how to model n-ary relationships.
4. Properties between classes and datatypes (**attributes**): If the target of a property is a data value, attributes are normally created to indicate that the expected value of the property belongs to a datatype (for example, integer, double, string, etc.).
5. **Property hierarchies**: A property r is subproperty of another property t if and only if every time the relation r holds between and individual and another individual or a value the property t also holds.
6. Additional **Axioms**: Other restrictions could be added to ontology elements, for example: cardinality constraints, property characteristics, universal or existential constraints, etc.

For more guidelines about ontology conceptualization and common modeling decisions we refer readers to the well-known "Ontology Development 101: A Guide to Creating Your First Ontology" (Noy and McGuinness, 2001), even though it was developed for ontologies following the frames paradigm, most of the recommendations are applicable directly or can be adapted to OWL ontologies. For advanced considerations concerning classes and individuals, we recommend reading Welty and Ferrucci's paper dealing with this issue (Welty and Ferrucci, 1999).

Tip: Ontology conceptualization

It is important to take into account that the conceptualization activity could be carried out on paper or on blackboard from where photographs of the conceptualization diagrams are taken to generate digital drawings and keep track of the conceptualization results. Another option is to use digital drawing tools, ideally with synchronous collaborative edition capabilities, and to use the evolving diagrams to keep history of the model and track decisions taken. Such diagrams can be reused during the documentation activity and could be used as communication tool to clarify the model with domain experts.

3.2.2. *Ontology encoding*

The goal of the encoding activity is to produce an ontology in an implementation language, for example, OWL (Hitzler et al., 2012). It is worth mentioning that along this work we refer to OWL as a superset

[11] https://chowlk.linkeddata.es/notation.html.

[12] http://www.w3.org/TR/swbp-n-aryRelations.

of OWL and RDF(S) constructs even though specific constructs are defined in the RDF(S) namespace and others in the OWL one. The ontology code resultant from this activity includes in addition to the ontology classes, properties and axioms, metadata, such as creator, title, publisher, license and version of the ontology in addition to metadata for each of the ontology terms. In the simplest scenario, the ontology development team would replicate and extend, if needed, the conceptualization model in an ontology language. This step is commonly done by using ontology editors and the developers would use them to introduce classes, relationships, attributes identified in the conceptualization as well as more detailed axioms and metadata.

In other cases some parts of the encoding might be automated or semi-automated. For example, it might be needed to transform a concept hierarchy into OWL or a thesaurus into a SKOS (Miles and Bechhofer, 2009) model or other types of terminologies or semi-structured resources into OWL structures. Detailed descriptions and examples of methods for non-ontological resources reuse for building ontologies are available in Villazón-Terrazas (2012).

Even though versioning conventions should be established for all artefacts being produced during the project, this practice has not yet been broadly adopted in ontology development projects. Therefore, it is advisable to establish a versioning convention at least to identify the different ontology versions produced along the project. For doing so, the ontology development team should define a versioning convention taking as inspiration the conventions used in software development.[13] Again, in the best case scenario the versioning convention should be applied to all possible artefacts, for example, set of requirements, conceptualization, test suite, etc.

Note that the encoding activity is closely related to the ontology reuse one, detailed in next sub-section.

3.2.3. Ontology reuse

Ontology reuse refers to the process of using available ontological resources for solving different problems (Suárez-Figueroa et al., 2015). If an ontology O_1 reuses another ontology O_2, it is possible to express statements and queries using terms of both of them. Therefore, this reuse is a factor in favor of the interaction between systems that assume O_1 and systems that assume O_2. Moreover, it can facilitate posing a federated query to a data source expressed according to O_1 and a data source expressed according to O_2. These two facts can be summarized saying that ontology reuse eases interoperability.

The reusability potential of ontologies and its benefits have been one of the main claims for ontology development since the first studies in the ontological field (Gruber, 1993; Neches et al., 1991, Studer et al., 1998, etc.). This vision and argument has driven research in different directions, for example: a) developing ontology modularization techniques to ease ontology reuse (d'Aquin et al., 2007; d'Aquin, 2012); b) proposing ontology design methodologies based on patterns reuse (Presutti et al., 2012); c) developing ontology registries where ontologies are published regardless the applications in which they are meant to be used and the data they are supposed to annotate (d'Aquin and Noy, 2012), for example Linked Open Vocabularies (Vandenbussche et al., 2017) or BioPortal (Noy et al., 2009); or d) analyzing the way practitioners actually carry out the ontology reuse activity (Poveda-Villalón et al., 2012; Katsumi and Grüninger, 2016).

As presented in Poveda-Villalón et al. (2012), ontology reuse is often not performed by means of importing the reused ontologies (hard reuse), but just by referencing the URIs of the reused ontology elements (soft reuse). Fig. 9 represents the following different types of reuse extending the cases presented in Poveda-Villalón et al. (2012). In the figure, four ontologies are represented (`onto1`, `onto2`, `onto3` and `onto4`). The different types of reuse are explained from `onto1` perspective, that is, how `onto1` reuses the other ontologies.

[13] https://semver.org/.

- **Hard reuse**:
  - **How to implement it**: This type of reuse is implemented by the OWL construct `owl:imports`. In this case the reused ontology is reused as it is, and as a whole. As the `owl:imports` construct is transitive, the imported ontologies by the one being reused will be also part of the ontology being built. This type of reuse is equivalent to the *as_is* type described by Katsumi and Grüninger (2016) and to the type "as a whole" proposed by Suárez-Figueroa et al. (2015). The *extension* proposed in Katsumi and Grüninger (2016) could be applied after a hard reuse step.
  - **Example of hard reuse in** Fig. 9: `onto1` reuses `onto2`, following the hard reuse type, by means of the `owl:imports` statement within the metadata dotted box in `onto1`. It can be observed that `onto2` is represented within `onto1` as consequence of the import mechanism.
  - **Points to consider**:
    * A whole ontology can be incorporated into another one with only one statement of the `owl:imports` construct.
    * The updates in the reused ontology are reflected in the ontology importing the reused one when reusing the ontology IRI instead of a particular version IRI.
    * Currently, there is no notification of modifications from the reused ontology but the changes are reflected in the ontology importing another one.
    * If the imported ontology becomes unavailable on the web, the ontology importing it will lose the statements of the imported one.

- **Soft reuse**: this type of reuse is implemented by referring to other ontology elements URIs.
  - **How to implement it**: This type of reuse is implemented by referring to other ontology elements URIs. It could also include the reproduction of some parts of the reused ontology in the reusing one. This type of reuse could be applicable to carry out the *extraction, extension and combination* types described by Katsumi and Grüninger (2016).
  - **Examples of soft reuse in** Fig. 9:
    1. References in `onto1` to elements defined in other ontologies: `onto3:ClassC0`, `onto3:ClassC03`, `onto4:ClassD04`.
    2. Generalization of referenced elements: `onto3:ClassC03` defined as subclass of `onto1:ClassA0` in `onto1`. The same could be done for specializing a term.
    3. Reproduction of a subpart of the ontology: `onto3:ClassC03` defined as subclass of `onto3:ClassC0` in `onto1` as originally defined in `onto3`.
    4. Replicate an element and keep provenance to original definition: declaration in `onto1` of `onto1:ClassA04` and link through an equivalent class statement to `onto4:ClassD04`. This reuse could appear in several situations, for example, the reused ontology element has been discovered by the developers after creating the ontology linking to it; the ontology development team aims at minimizing the number of namespaces to deal with, then they create every element in the owned namespace and link to external ontologies to allow interoperability by following the mappings; among other cases.

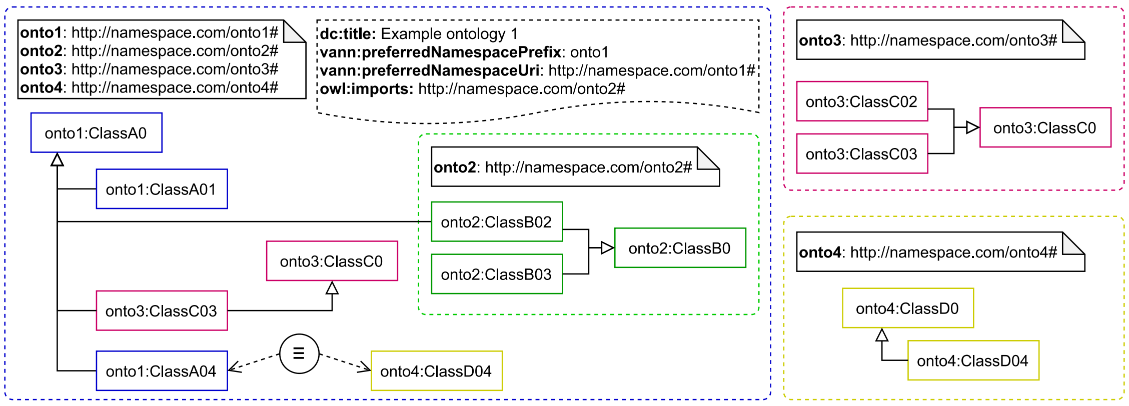

onto1: http://namespace.com/onto1#
onto2: http://namespace.com/onto2#
onto3: http://namespace.com/onto3#
onto4: http://namespace.com/onto4#
dc:title: Example ontology 1
vann:preferredNamespacePrefix: onto1
vann:preferredNamespaceUri: http://namespace.com/onto1#
owl:imports: http://namespace.com/onto2#
onto1:ClassA0
onto1:ClassA01
onto2: http://namespace.com/onto2#
onto2:ClassB02
onto2:ClassB03
onto2:ClassB0
onto3:ClassC0
onto3:ClassC03
onto1:ClassA04
≡
onto4:ClassD04
onto3: http://namespace.com/onto3#
onto3:ClassC02
onto3:ClassC03
onto3:ClassC0
onto4: http://namespace.com/onto4#
onto4:ClassD0
onto4:ClassD04

Fig. 9. Soft and Hard reuse types.

– **Points to consider**:
  * The reused parts of other ontologies remain stable in the ontology being built even though there are changes in the original one.
  * The ontology being built might lose track to definitions in the reused ontology when it evolves.

Another way of reusing ontological resources according to the NeOn methodology is by reusing Ontology Design Patterns (ODPs): reusable successful solutions to recurrent modeling problems known as ontology design patterns (Presutti et al., 2012). The Ontology Design Patterns portal[14] provides a catalogue of patterns for several tens of domains (academy, agriculture, biology, parts and collections, etc.) and Presutti et al. (2012) provides guidelines for the design of ontologies based on ODPs.

The ontology reuse activity may be taken into account along the following activities:

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 activity 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 activity, 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.

It should be taken into account that even though there are many benefits claimed about ontology reuse, such an activity is not cost free and some common issues might arise when trying to reuse ontologies (see Q4 in Section 1). In fact, Fernández-López and colleagues (Fernández-López et al., 2019) carried out a study on LOV ontologies, and they identified the following difficulties in ontology reuse, for example: heterogeneity in the natural language used, heterogeneity between the concepts of the reusing and the reused ontologies, deficiencies in the documentation, loss of information due to an imported ontology that is not available, unavailable license, etc.

The scope of this ontology reuse section is to provide some general guidelines for reusing ontologies. More extended guidelines can be found in Carriero and colleagues' work (Carriero et al., 2020) and in Fernández-López and colleagues' work (Fernández-López et al., 2012).

Tip: Ontology reuse

The ontology reuse activity might be carried out in line with the ontology encoding one, that is, once the model is stable, look for ontologies that already represent such model. In this sense, when encoding the conceptualized model, ontology developers should search for existing ontologies in order to reuse them as a whole or reuse part of them. In this case, the conceptualization might be subject of changes when analyzing existing models.

For experienced developers, the reuse activity could be carried out both during the conceptualization or the encoding activities. In those cases in which the development team is knowledgeable about existing ontologies in the domain at hand, they can drive the conceptualization based on standards or well-known ontologies, for example considering patterns defined in W3C SSN/SOSA or ETSI SAREF ontologies when developing ontologies for sensors and actuators.

From our experience, we discourage looking for ontologies exhaustively from the ontology requirements without having a more clear idea of the conceptualization, as it can be very time consuming to study, understand and compare the many models that can arise from searching from the terms in the requirements. The most convenient situation is starting the conceptualization without looking for many ontologies (but considering those that the development team is knowledgeable about) and looking for ontologies once a first conceptualization is stable but open to modifications according to the ontologies to be reused.

3.2.4. Ontology evaluation

This activity 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). Two compilations about ontology evaluation techniques, methods and tools are listed and described in Sabou and Fernandez (2012) and McDaniel and Storey (2019).

[14] http://ontologydesignpatterns.org

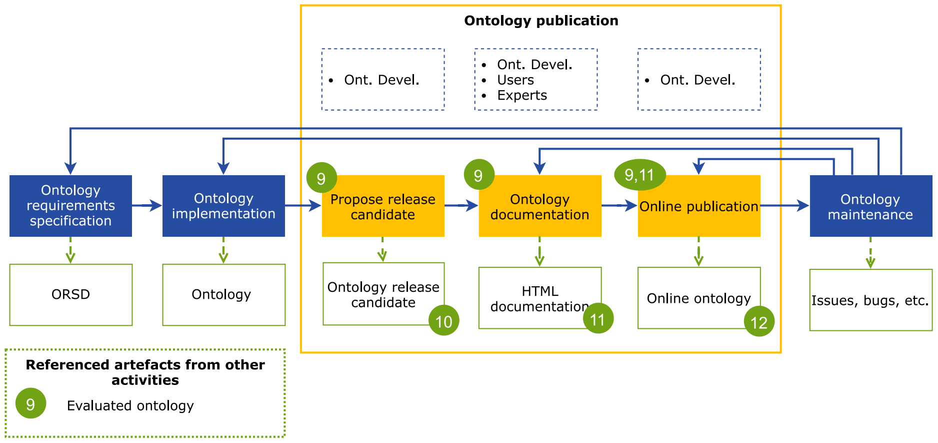


Fig. 10. Workflow proposed for ontology publication.

Functional requirements can be formalized into test cases to verify that they are satisfied by the ontology. These tests 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 CQ together with the expected result of the query. Instead of adding SPARQL queries, test expressions following the testing method defined in Fernández-Izquierdo and García-Castro (2018) can also be added and stored in RDF files following the Verification Test Case ontology.[15]

According to the evaluation results, the ontology development team could go back to the conceptualization or encoding activities in order to fix bugs or improve the current version of the ontology.

Tip: Ontology evaluation

From our experience, it has been observed that when there are domain experts involved in the ontology development process and the ontology is not a part of a software system, the execution of SPARQL queries or test expressions in order to gather data based on ontology requirements helps domain experts to easily identify misunderstandings and lack of knowledge described in an ontology.

When ontologies are part of a software product or when only ontology developers are involved in the ontology development process, ontology verification and data coverage analysis are common practices for evaluating the ontology.

3.3. *Ontology publication*

The aim of the ontology publication activity 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-activities as shown in Fig. 10.

It should be mentioned that not in every ontology development project the ontology is intended to be published on line; however as the LOT methodology is oriented to the Linked Open Data and Semantic Web paradigms, this activity is highly recommended, so that the produced ontology can be reused by others. However, the decision is on the development team or project characteristics, and if the ontology is not published online, the “Online publication” sub-activity could be omitted while the previous two (“Propose release candidate” and “Ontology Documentation”) are recommended to share the ontology internally within the project at hand.

[15] https://w3id.org/def/vtc#.

3.3.1. *Propose release candidate*

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). In this case, the version at hand becomes a release candidate (see Fig. 10). In case the ontology is not selected as release, the ontology development team generates a pre-release version of the ontology. Both release and pre-release versions of the ontologies are evaluated and ready to be used.

3.3.2. *Ontology documentation*

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. According to the best practices for publishing FAIR (Findable, Accessible, Interoperable and Reusable) vocabularies and ontologies provided by Garijo and Poveda-Villalón (2020), this documentation includes:

- A human-readable description of the ontology, commonly as an HTML document, that describes the classes, properties, data properties and individuals of the ontology. The domain experts have to collaborate with the ontology development team to describe the classes and the properties; otherwise the ontology developers should look for sound descriptions of the terms included in the model. This information is normally included in the form of annotations in the OWL code of the ontology.
- Additionally, the HTML description of an ontology should contain metadata, such as the license URI, the title being used, the creator, the publisher, the date of creation, the last modification or the version number. This information associated to the ontology is important to provide an overview and identify an ontology, understand its usage conditions and its provenance. This information is normally included in the form of annotations in the OWL code of the ontology.
- The HTML documentation should also contain information oriented to human consumption about the intended use of the ontology, its purpose and scope. It is encouraged to add an abstract and detailed descriptions of the model that help user to understand better the ontology.
- Diagrams which store the graphical representation of the ontology, including taxonomy and class diagrams. Garijo and Poveda-Villalón (2020) propose a notation for representing classes and properties based on the UML_Ont profile (Haase et al., 2009).

• Examples of use that illustrate how to use ontologies in practice.

Additionally, the documentation of the ontology can also include the following artefacts generated during the development process:

• The list of requirements identified during the requirement specification activity and that should be satisfied by the ontology.
• The list of test cases used during the ontology evaluation activity to verify the ontology.

3.3.3. *Online publication*

Once the documentation of the ontology has been generated, we proceed to the online publication step. First of all, the ontology is published on the Web following the practices described for publishing vocabularies on the Web provided by the W3C,[16] so that it is accessible via its URI as a file in a formal language and as human-readable documentation using content negotiation.

Tip: Ontology documentation and publication

From our experience, we observed that is it very useful to include within the HTML documentation graphical representations of ontology instantiations and examples of use, together with the ontology diagrams. These materials allow domain experts and users to better understand the domain described in the ontology as well as its structure and usage.

Note that while one might be tempted to generate as much documentation as possible during the conceptualization and implementation (for example adding labels and comments for the defined terms) to be reused during the documentation activity, in some cases a number of modifications are needed after evaluating the ontology from a functional point of view. In this sense, it is recommended to accomplish some parts of the documentation (for example the examples generation) after the evaluation activity.

3.4. *Ontology maintenance*

The goal of this activity is to update the ontology during its life cycle. This may be needed due to different situations as shown in Fig. 11. In this case, the internal boxes shown in Fig. 11 represent potential situations that may occur, and that the ontology development team should react to, rather than activities that the ontology development team should carry out. These situations are **bug detection**, which also include suggestions and improvements, and **new requirements** identification. In both cases, the involved actor (developer, user, etc.) detecting the bug or requirement should report it to the ontology development team. If the ontology is published online and open to the community, these activities could be triggered by community members involved in a given domain.

This activity may be triggered during the ontology development process in which the new requirements implementation or bug fixing would be scheduled in one or more sprints. This activity may be also triggered after the ontology development process in which a new version or revision of the ontology should be generated.

Tip: Ontology maintenance

Monitoring and maintaining the traceability of the requirements, bugs and suggestions identified in this activity allows storing discussions and decisions taken that can be later reused. For this reason, it is advisable to use issue tracker systems that allow issue management and discussion, possibly in an open environment so that external feedback can be received.

[16] http://www.w3.org/TR/swbp-vocab-pub/.

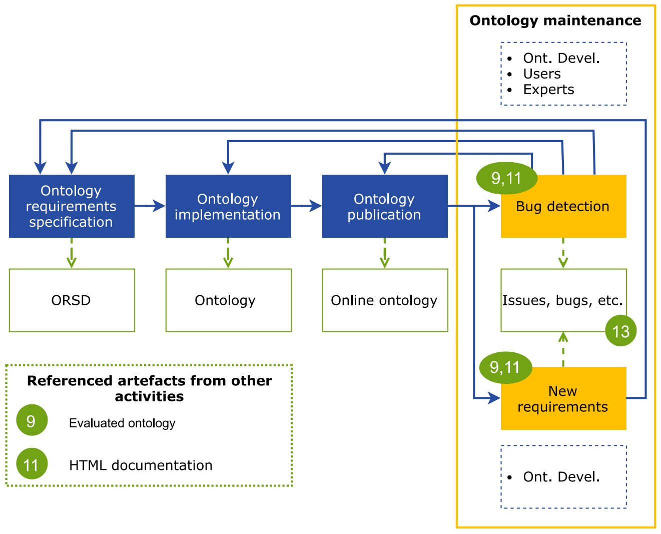


Fig. 11. Workflow proposed for ontology publication.

4. Software support

In order to support the activities described in Section 3, this section presents several tools and technologies oriented to ontology development (see Q6 in Section 1). In the first place, each subsection from 4.1 to 4.3 includes references and description of specific tools and methods that can support in particular each ontology development activity. Then, technologies commonly used in software development that can support ontology development are briefly described in Section 4.5. The relation between tools supporting the methodology and each activity is depicted in Section 4.6.

4.1. *Software support for requirement specification*

One of the options **to store and manage the Competency Questions** or natural language statements resulting from the requirements specification task is using collaborative spreadsheets,[17,18] which allows several participants to collaborate in the writing of the specification document. See that this activity also generates as output the ORSD document. The LOT methodology provides a template for such document both in docx and LaTeX formats.[19] Note that this document can be included in the git repository of the ontology in order to be available for the rest of participants involved the development process.

When ontological requirements are extracted following the METHONTOLOGY alike tables, tools that allow managing tabular information in a collaborative way are recommended. During the BIMERR project, Confluence[20] pages were designed to collaboratively edit the information that needed to be represented (see Figs. 4–6).

4.2. *Software support for implementation*

In practical terms, the **ontology conceptualization** is materialized by the logical descriptions of the ontology that could be represented by a set description logic formalization or graphical descriptions. From our experience, the most common method for ontology conceptualization is to draw diagrams of the concepts, relations and axioms (to a certain

[17] https://spreadsheets.google.com.
[18] An example can be found on the https://docs.google.com/spreadsheets/d/1_VcoGD5Qq6iKr8-XNGJGsOo475aiZsnKf0i01awdZkc/edit?usp=drive_web&ouid=103242980179942151298.
[19] A template is available at https://github.com/oeg-upm/LOT-resources.
[20] https://www.atlassian.com/software/confluence.

extend). The conceptualization of an ontology can be carried out in a blackboard or in a sheet of paper or using digital tools. For free-style diagrams, one could use general purpose tools as diagrams.net[21] or yEd[22] among others.

Another alternative is to generate the conceptualization using systems specifically designed for ontology development. In this case, we can find several options each of them are based in a particular notation oriented to the transformation of the diagrams into OWL code, as for example:

- **Chowlk**: An on-line application[23] (Chávez-Feria et al., 2021) 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.

  Notation: The Chowlk notation.[24] is based on the UML_Ont profile proposed in Haase et al. (2009) This notation defines building blocks for each OWL 1 elements as well to insert ontology metadata and prefixes. For most of the building blocks, the Chowlk notation offers different alternatives for more compact or detailed representations.

- **WebVOWL**: 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 was merged within WebVOWL. The system could be used online[25] or downloaded for local installation.

  Notation: It is based on the VOWL visual notation (Negru and Lohmann, 2013) for OWL ontologies, that is oriented to semantic networks in which concepts are represented by circles and object properties and datatype properties are represented by arrows.

- **OWLGred**: A graphical editor for OWL ontologies[26] 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[27] or downloaded for local installation.

  Notation: The OWLGred editor is based on the UML class diagram notation.

- **Graffoo**: Graffoo[28] 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.

  Notation: Graffo notation defines classes and class restrictions as rectangles, datatypes and datatype restrictions as rhomboids, individuals as circles and assertions, annotation properties and datatype and object properties as arcs.

Another tool for ontology conceptualization is Gra.fo,[29] 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. There is also a plugin for Protégé (Musen, 2015),[30] named OWLAx (Kamruzzaman et al., 2016), which provides an interface for drawing class diagrams, and a command to generate axioms from the given diagram.

The ontology can be **implemented** by means of an ontology editor, such as Protégé, which is an open-source editor that allows installing plugins to add new functionalities to the tool, or its online version WebProtégé (Horridge et al., 2019).[31] Protégé could be also used during the implementation activity to complete a refine preliminary OWL versions of the ontology generated with ontology conceptualization tools (mentioned above) with limited edition capabilities. Another tool for implementing ontologies is the Fluent Editor,[32] which allows editing and manipulating complex ontologies using Controlled Natural Language. TopBraid Composer[33] is a commercial tool for ontology implementation that also offers additional functionalities, such as the import of databases, XML-Schemas or UML. PoolParty[34] (Schandl and Blumauer, 2010) is a web-based tool to create collaboratively SKOS (Simple Knowledge Organization System) and RDF vocabularies.

As mentioned in Section 3.2, the scope of this paper regarding ontology resuse is to provide some general guidelines. In this sense, to help ontology developers during the **ontology reuse activity**, several ontology registries were proposed. Some of these registries are oriented to a general purpose, such as Linked Open Vocabularies (LOV)[35] (Vandenbussche et al., 2017), Ontohub[36] (Codescu et al., 2017) and BARTOC.[37] Other registries are focused on specific domains, such as Linked Open Vocabularies for Internet of Things (LOV4IoT)[38] (Gyrard et al., 2016) for the Internet of Things domain, Vocabularies for Open Data for Cities[39] for the smart cities domain, BioPortal[40] (Noy et al., 2009) and OntoBee[41] (Ong et al., 2017) for the biomedicine domain, and AgroPortal[42] (Jonquet et al., 2018) for the agriculture one.

Regarding the **evaluation activity**, ontologies can be partially evaluated using reasoners such as Pellet (Sirin et al., 2007), or Hermit.[43] For evaluating ontologies regarding modeling issues, OOPS! (OntOlogy Pitfall Scanner!)[44] (Poveda-Villalón et al., 2014) is a web application and web service for ontology diagnosis that automatically detects the most common pitfalls that appear when developing ontologies, including semantic and structural checks as well as best practices verification (see Q7 in Section 1). Concerning ontology verification, Themis[45] (Fernández-Izquierdo and García-Castro, 2019a) is an online tool also available as a REST (Representational State Transfer) API (Application Programming Interface) that allows executing tests associated to the requirements collected during the requirements specification activity. It proposes a testing language based on lexico-syntactic patterns, aiming at facilitating the definition of tests. TDDOnto2 tool[46] (Davies et al., 2019) is another Protégé plugin for Test-Driven Development. OntoCheck (Schober et al., 2012) is a plug-in for Protégé for evaluating ontologies. It helps to clean up an ontology with regards to its lexical heterogeneity and allows verifying whether a particular orthographic or morphosyntactic naming convention is fulfilled.

[21] https://www.diagrams.net/.
[22] https://www.yworks.com/products/yed.
[23] https://chowlk.linkeddata.es/.
[24] https://chowlk.linkeddata.es/notation.html.
[25] http://vowl.visualdataweb.org/webvowl.html See that the edition capabilities are currently experimental
[26] http://owlgred.lumii.lv/.
[27] http://owlgred.lumii.lv/online_visualization.
[28] https://essepuntato.it/graffoo/.
[29] https://gra.fo/.
[30] https://protege.stanford.edu/.
[31] https://webprotege.stanford.edu/.
[32] https://www.cognitum.eu/Semantics/FluentEditor/.
[33] https://www.topquadrant.com/products/topbraid-composer/.
[34] https://www.poolparty.biz/ontology-management/.
[35] https://lov.linkeddata.es.
[36] https://ontohub.org/
[37] https://bartoc.org/.
[38] https://lov4iot.appspot.com/.
[39] http://vocab.linkeddata.es/datosabiertos/.
[40] https://bioportal.bioontology.org/.
[41] http://www.ontobee.org/.
[42] http://aims.fao.org/agroportal.
[43] http://www.hermit-reasoner.com.
[44] http://oops.linkeddata.es/.
[45] http://themis.linkeddata.es/.
[46] https://github.com/agnieszkalawrynowicz/tddonto.

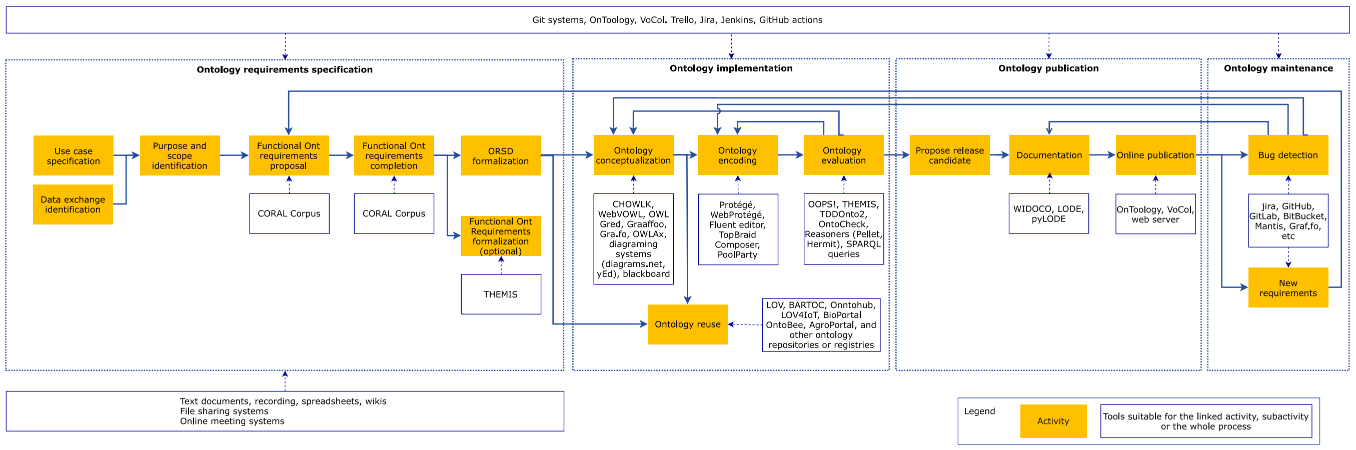

Git systems, OnToology, VoCol. Trello, Jira, Jenkins, GitHub actions
Ontology requirements specification
Use case specification
Data exchange identification
Purpose and scope identification
Functional Ont requirements proposal
CORAL Corpus
Functional Ont requirements completion
CORAL Corpus
ORSD formalization
Functional Ont Requirements formalization (optional)
THEMIS
Ontology implementation
Ontology conceptualization
CHOWLK, WebVOWL, OWL Gred, Graaffoo, Gra.fo, OWLAx, diagraming systems (diagrams.net, yEd), blackboard
Ontology encoding
Protégé, WebProtégé, Fluent editor, TopBraid Composer, PoolParty
Ontology evaluation
OOPS!, THEMIS, TDDOnto2, OntoCheck, Reasoners (Pellet, Hermit), SPARQL queries
Ontology reuse
LOV, BARTOC, Onntohub, LOV4IoT, BioPortal OntoBee, AgroPortal, and other ontology repositories or registries
Ontology publication
Propose release candidate
Documentation
WIDOCO, LODE, pyLODE
Online publication
OnToology, VoCol, web server
Ontology maintenance
Bug detection
jira, GitHub, GitLab, BitBucket, Mantis, Graf.fo, etc
New requirements
Text documents, recording, spreadsheets, wikis
File sharing systems
Online meeting systems
Legend
Activity
Tools suitable for the linked activity, subactivity or the whole process

Fig. 12. Tools supporting ontology development activities.

4.3. *Software support for publication*

The development team can use several tools for **documenting ontologies** before their publication (see Q8 in Section 1). Live OWL Documentation Environment (LODE) (Peroni et al., 2012) is an open-source service that automatically extracts ontology elements and renders them as ordered lists, together with their textual definitions, in a human-readable HTML page designed for browsing and navigation by means of embedded links. WIDOCO[47] (Garijo, 2017) is a wizard for documenting ontologies that also guides users through the steps to be followed when documenting an ontology, indicating missing metadata that should be included. WIDOCO takes as input an ontology to generate a set of linked HTML pages containing a human readable description of the ontology. It integrates and extends well established tools, like LODE for term documentation, WebVowl (Lohmann et al., 2014) for graph visualization, Bubastis (Malone et al., 2010) for adding change logs and Licensius[48] for completing ontology license metadata.

The HTML documentation can be enriched with ontology diagrams obtained from the conceptualization task using the previously mentioned drawing tools (see Section 4.2). Moreover, these drawing tools can be used to generate examples of use of the ontology to improve the coverage of the documentation and therefore the understandability of the ontology.

Once the ontologies are documented should be **published** on-line following the best practices. This task, which could be seen as the ontology deployment, could be carried out manually by the system administrators of the web domain where the ontology is going to be published, or could be aided by some systems. OnToology[49] (Alobaid et al., 2019) is an online tool integrated with GitHub that generates the HTML documentation and evaluates the ontology. OnToology allows users to reserve a name for their ontology with a permanent URI following the scheme <https://w3id.org/def/<user-chosen-name> and uses the w3id services to point to the location of the published ontology and its resources with content negotiation. Another tool supporting ontology publication is VoCol (Halilaj et al., 2016). However, VoCol does not allow users to decide on a naming scheme when publishing their ontologies and it cannot assign permanent identifiers to an ontology.

4.4. *Software support for maintenance*

For the maintenance of ontologies, the development team can use ontology development oriented tools that also support tracking issues and communication between developers. This is the case of WebProtégé, which supports threaded notes and discussions and allow issues to be tracked and associated with terms and change sets. Moreover, developers can "watch" ontologies for changes that affect a specific term or hierarchy. Gra.fo also supports the maintenance of ontologies, providing support for adding comments related to concepts, attributes, relationships and specializations. Moreover, it allows tracking the ontology history, name a version and restore to any previous version. Finally, LOT methodology also recommends using GitHub, or other git systems, for ontology maintenance since it serves as an online infrastructure to maintain ontology versions as they are being developed and it provides an issue tracker to open issues and maintain discussions about the development.

4.5. *Adoption of software development tools in ontological engineering projects*

To manage ontology development processes it is needed to considered several aspects, such as version control, traceability of ontology artefacts, changes integration and task management. Inspired by software engineering practices, which are widely used in software projects, this section proposes a set of tools to support ontology projects management.

For supporting the control of **ontology changes** or the changes made to the other artefacts generated during the ontology development process, a version control system is required to keep track of every modification made to them. By capturing and storing the changes, it allows going back to previous versions of the ontology or its artefacts if a mistake is made. The version control systems also ensure conflict resolution and allow the integration of conflicting changes. One of the main systems used for version control in software projects is Git,[50] which allows maintaining and synchronizing both local and remote repositories, provides support for collaborative development and eases the consistency of the different products of the development process. Github,[51] GitLab[52] and Bitbucket[53] are examples of hosts for remote Git repositories. The LOT methodology recommends using Git version control systems as supported by the literature review presented in Kotis et al. (2020) showing that ontology developers frequently use freely available tools such as GitHub for easing their effort when managing versions of ontologies, and also for supporting distributed collaboration in ontology developments. Apart from the benefits provided by Git systems, when software developers are involved in ontology development

[47] https://github.com/dgarijo/Widoco/.
[48] http://licensius.com/.
[49] https://ontoology.linkeddata.es/.
[50] https://git-scm.com.
[51] https://github.com.
[52] https://gitlab.com.
[53] https://bitbucket.org.

projects, they can access the ontologies that they need in a way that is familiar for them. In particular, LOT recommendation is to use GitHub repositories as they can be supported by OnToology, which is an online tool that integrates different ontological development activities such as ontology documentation, evaluation and publication. A similar service is VoCol, which is designed as a tool to help collaborative vocabulary development and use Git repositories to maintain vocabulary-related files. VoCol provides support for evaluation, documentation, visualization and publication of ontologies. However, while OnToology is an online tool that does not require any installation setup, VoCol requires to install the VoCol environment on a local machine or a web server.

For managing ontology development processes, it is also required to **keep track of the tasks and status of the project**. Inspired by software development practices, this operation can be performed in a Kanban-based way. In this sense applications as *Trello*[54] and *Jira*[55] could be reused.

Finally, to continuously integrate changes as well as to continuously apply quality control (e.g., validating and verifying the ontology), **continuous integration** tools can be used. *Jenkins*[56] and *Travis*[57] are tools that can be integrated with Git repositories to support ontology development. An example of the adaptation of the continuous integration techniques to ontology development is the SAREF pipeline.[58]

Tip: Ontology repositories in version control systems

If several ontologies are developed in the same project, all of them can be included in just one Git repository (e.g. former SAREF extension development), or one Git repository per ontology can be used (e.g., BIMERR and VICINITY ontology networks). If all the ontologies are developed at the same time by the same working group, the first option could be more comfortable for the development team, who do not have to maintain and update several repositories. Nevertheless, in the second case, each repository can be browsed focussing in its ontology. Besides, if one of the working groups is interested just in one ontology, they can download it without downloading the rest of the ontologies. However, it is recommended to keep different ontologies in different Git repositories to support independence of the releases.

4.6. Summary of supporting tools for ontology development

This section provides a graphical alignment (Fig. 12) between the tools mentioned in previous sections and the (sub)activities proposed by the LOT methodology (see Section 3. Such figure also includes tools that can be used along the project.

5. Validation

As de Hoog (1998) asserted, "It is extremely difficult to judge the value of a methodology in an objective way. Experimentation is, of course, the proper way to do it, but it is hardly feasible because there are too many conditions that cannot be controlled". On the one hand, the introduction of an experimental toy problem would violate the basic assumption behind the need for methodological guidelines: that the development process be complex. On the other hand, if we apply the argument provided by de Hoog for knowledge-based systems to the ontology development field, we can observe that it is unlikely that an ontology development project would be funded more than once to use different methods and tools each time in order to compare results.

Table 1 has been created to analyze the coverage of LOT in comparison with the methodologies mentioned in Section 2 based on the needed aspects in Ontology Engineering mentioned in Tudorache's work (Tudorache, 2020). All these mentioned methodologies describe the set of activities that should be carried out in any ontology development process, including the encoding of the ontology and only in some cases the support activities such as the maintenance involving the community behind an ontology domain. Special attention needs the publication activity which is not provided by previous methodologies but is critical considering that ontologies in the Semantic Web are meant to be shared online[59] and the current emergence of FAIR vocabularies publishing as identified in Hugo et al. (2020), Poveda-Villalón et al. (2020). Moreover, the majority of these existing methodologies lack guidelines for ontology reuse and none of them mentioned potential tools that can be helpful to ontology developers or the tools mentioned are deprecated and do not follow agile approaches aligning with software development. It should also be mentioned that the ontology consumption is not covered by any methodology.

Moreover, the use of the methodology was validated in multiple projects (see Q5 in Section 1), which are described in Appendix,[60] showing that LOT has the following features:

1. It has been used in projects where ontologies with hundreds of entities have been defined.
2. The activities proposed along this paper have been actually carried out in the projects using the proposed methodology with a technological support.
3. The projects where the framework has been applied are diverse: different domains, different natural languages, different purposes, different team configurations and different project types (different funding schemes).

5.1. Discussion

This section aims at providing some clarifications, observations and discussion about the proposed methodology taking as basis the experience of its application in the projects above-mentioned. As shown in Tables 2 and 3 (See Appendix) the methodology has been successfully applied in 18 projects (4 ETSI standardization projects, 4 European projects, 4 Spanish national/regional projects, 2 French regional projects, 4 projects with private companies) with different profiles involved. It should be mentioned that the authors of this paper were not involved in some of these projects, such as the one related to the BTN100 ontology, datos.ign.es, Ciudades Abiertas and D2KAB projects. Moreover, the authors of this paper had only support roles in the development of the Noise ontology and the CASO & IRRIG ontologies.

We can observe a tendency to use collaborative systems for ontology requirements gathering and project artefacts management. This represents a need due to the geographical situation of team members in international projects but also a convenience solution for local projects. However, there are subtle differences between the type of collaborative systems used. More precisely, for the activities previous to the encoding one, systems that allow synchronous edition are commonly used, while from the ontology encoding activity onwards no collaborative tools are used (Protégé, OOPS!, Widoco, reasoners, etc.) but the generated artefacts are shared through collaborative environments, such as git systems. The exception to this rule is the use of diagrams.net in the documentation activity; however the use of diagrams at this point has been in general related to the reuse of the ontology conceptualization diagrams during the documentation. This situation might be motivated

[54] https://trello.com.
[55] https://www.atlassian.com/es/software/jira/features.
[56] https://www.vogella.com/tutorials/Jenkins/article.html.
[57] https://travis-ci.org/.
[58] https://portal.etsi.org/STF/STFs/STF-HomePages/STF578.
[59] https://bvatant.blogspot.com/2012/02/is-your-linked-data-vocabulary-5-star_9588.html.
[60] Some of the projects where the LOT methodology has been applied are presented in Tables 2 and 3 in Appendix where the different characteristics of each project are detailed. To facilitate reading and processing the tables, an excel file containing the same information is provided at https://lot.linkeddata.es/data/LOTprojects.xlsx. Such file will be uploaded to Zenodo for the final version of the paper.

Table 1
Methodologies' characteristics comparison based on Tudorache (2020). (Depr: tool support deprecated).

	Methontology	On-To-Know ledge	Diligent	NeOn	eXtreme method	XD	RapidOWL	Samod	Upon lite	LOT
Ontology development workflow	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ontology reuse				✓						✓
Ontology evaluation	✓	✓		✓	✓	✓		✓		✓
Ontology maintenance through community								✓		✓
Tooling	Depr			Depr						✓
Ontology consumption										

by the fact that online collaborative ontology editors are not as robust or usable as desktop ones.

Another evidence of the success of the git-based approach for managing all artefacts during the ontology development project is the recent adoption of a similar procedure within the new SAREF pipeline for developing and maintaining extensions reported in ETSI (2020).

One important observation about SAREF extensions is the fact that in some cases domain experts were not directly involved in the ontology development. However, during the SAREF4CITY development some workshops to gather stakeholder comments were organized, in some occasion with the collaboration of the European Commission[61] involving the community around the smart cities domain.

Regarding the requirements specification techniques we can observe that while writing CQs has been the leading technique is the last decades, it is in general being complemented with other techniques like the description of use cases. In addition, we can observe that the CQs technique is not used in the projects presented when tables are used to write down requirements (see DELTA and BIMERR) or the CQs are used in addition to the tables (VICINITY). Note that in these three projects there were software developers involved who are normally comfortable with structured data. Also in this line, it should be clarified that for the MASS MEDIATOR project the ontology developer was the same person developing the API for which the ontology was built, therefore the requirements were not elicited nor documented in standard ways.

In 12 out of the 18 projects, data sources, data schemas or API descriptions have been taken as knowledge sources. This fact might imply that data integration and data-driven ontology developments are every day more common.

Finally, it is worth noting that in every project UML based notations have been used for the conceptualization activity. In those projects involving software developers it eases the communication with them as they are used to such representations. For the rest of the projects, as there was no preferred diagramming notation by the domain experts, the UML_Ont-based notation (that is the previous version of the Chowlk notation) was chosen to facilitate the transition to the OWL implementation.

6. Conclusions and future work

Along this paper the LOT methodology, together with recommended tools and practical tips stemming from decades of experience in ontology engineering and use of existing methodologies that served as basis for its development, has been presented. Throughout the paper, the questions opening the introduction have been answered. The LOT methodology has evolved during the last decade (since it inception (Poveda-Villalón, 2012) and its evolution (García-Castro et al., 2017)) and it is intended to evolve with feedback from its usage. To this end a GitHub repository has been established not only to share resources associated with the methodology, but also to gather feedback from the community and contributions.

The presented methodology has been validated by comparison with the state of the art, showing that LOT covers activities that are not described in any other methodology, such as ontology publication, and that it also contributes with tips and recommendations based on the authors experience. LOT has also been validated by the successful application in 18 projects with very heterogeneous configurations (development team, funding scheme, application, domain,[62] data sources, etc.) including projects where the methodology authors have not been involved.

During the methodology and projects development, it has been observed the need for collaborative edition tools and communication systems to carry out some activities, mostly regarding requirements specification. On the one hand, in some cases general purpose collaborative environments (as wikis or more complete solutions such as confluence) could be used for particular ontology developments, which might convey a cost and configuration and maintenance effort. On the other hand, this need might open the opportunity for the development of new systems specifically designed for ontology projects.

It has also been observed that there is still a need for practical and detailed guidelines at least for the following activities: conceptualization (including ontology modularization), ontology reuse, ontology documentation including examples to facilitate their usage and requirements traceability. Our future lines in this regard involve generating guidelines for ontology conceptualization based on diagramming following the Chowlk visual notation. Another important point to broaden the methodological guidelines with regard to the ontology lifecycles is to address the post-development phase, that is, the exploitation of ontologies. In this sense, the generation of guidelines for ontology usage through SHACL shapes (Knublauch and Kontokostas, 2017) to validate that data follows ontology restrictions and the definition of APIs to consume data annotated with ontologies should be explored in the near future.

Finally, future work will involve the generation of guidelines for possible ontology development workflows including tools based on project configurations. For example, to suggest whether to use one or another tool or workflows depending on the ontology development team characteristics, the data sources and development time, taking as inspiration the project configuration shown in Tables 2 and 3 and the lessons learnt during such developments.

CRediT authorship contribution statement

María Poveda-Villalón: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Alba Fernández-Izquierdo:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Mariano Fernández-López:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Raúl García-Castro:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

[61] https://digital-strategy.ec.europa.eu/en/events/workshop-towards-harmonization-smart-city-models-interoperable-extension-saref-smart-cities.

[62] LOT is thought to be adopted by engineers. Therefore, it can be a contribution, not only in projects involving software engineers, but also engineers of other disciplines. An example is the application to Model-Based System Engineering (https://www.omgwiki.org/MBSE/doku.php?id=mbse:ontology)

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the project Técnicas y herramientas para la gestión de grafos de conocimiento para dar soporte a espacios de datos (Grant PID2020-118274RB-I00 funded by MCIN/AEI/ 10.13039/501100011033), funded by the Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación of Spain, by the BIMERR project funded from the European Union H2020 research and innovation programme under grant agreement no. 820621, the VICINITY project funded from the European Union H2020 research and innovation programme under grant agreement no. 688467 and by a Predoctoral grant from the I+D+i program of the Universidad Politécnica de Madrid. We also would like to thank to everyone that has followed the methodology in their developments and provided feedback and data, specially Paola Espinoza-Arias, Quang-Duy Nguyen and Catherine Roussey. All authors approved the version of the manuscript to be published.

Appendix. Example of projects developed using the LOT methodology

This Appendix describes several ontology development projects carried out following the LOT methodology including information about different characteristics of each project (name, team configuration and project type (national/regional, European, standardization, private)), as well as the methodological techniques and technological resources used in each activity are listed (Tables 2 and 3). Note that for some activities no methodological technique is mentioned, for example for maintenance or for ontology implementation as in all the cases it is done just by translating the conceptualization into OWL.

- **The SAREF ontology extensions**: The Smart Applications REFerence ontology (SAREF)[63] is a standard ontology supported and promoted by a collaborative effort between the European Telecommunications Standards Institute (ETSI) SmartM2M committee[64] and the European Commission. The first version of SAREF represented a reference ontology to be used by smart appliances, from lamps and consumer electronics to white goods like dishwashers,
regardless of their underlying communication protocols.[65] Since 2016, ETSI SmartM2M requested Specialist Task Forces to provide input on the management of SAREF and create SAREF extensions for several domains such as Agriculture (i.e., SAREF4AGRI), Industry and Manufacturing (i.e., SAREF4INMA) (de Roode et al., 2020), Smart Cities (i.e., SAREF4CITY), or Water (i.e., SAREF4WATR).
- **DELTA**: Due to the high variability in peak load electricity demand, Electricity System Operators are forced to have a capacity much higher than what is actually required so they can respond effectively to peak load spikes. DELTA European project[66] aims at building a framework that can respond to peak load demands for electricity across Europe, ensuring flexibility in response to peak demands for electricity as well as increasing the use of renewable power sources. The DELTA ontology is developed to provide interoperability among the different components involved in the project. Such ontology conceptualizes all the exchanged information, such as the energy price, the power profile of a system or the set of customers involved in the DELTA platform.
- **VICINITY**: The VICINITY European project[67] aims to allow for the discovery of Web Things and the interaction between them as if they belonged to a "social network of things". The devices are registered in the system in the same way as a person is registered in a social network. According to a series of parameters established by the responsible of the devices, these make visible their properties according to the privileges of each user. The domains needed to be modeled in the project were: Web of Things Devices (sensors, actuators and their properties) and mappings from the data sources to specific vocabularies.
- **BIMERR**: The goal of the BIMERR European project[68] is to design and implement a set of Building Information Modeling tools to support building renovations projects. The suite of tools involved in the project includes a decision support system that guides designers to select the best renovation scenario, applications to share annotation from different actors during the renovation process, laser scanners that automatically generate a 3D representation of the existing building, among others. Due to this high diversity of tools, the core of the system need an interoperability layer that support the flow of information between the applications. Such central data model is driven by a network of ontologies where each modules covers a specific domain within the building sector.
- **EasyTV**: EasyTV European project[69] faced the challenge of improving the access to mainstream multimedia products and services for people with different types and levels of disabilities, such as visually or hearing impairment. One of the modules developed is a crowdsourcing platform where user can share sign language videos in different languages where the videos representing the same concepts in different sign languages should be linked. These multilingual knowledge base of sign language is supported by an ontology used to annotate sign language videos and the data is stored in RDF.
- **The OTN ontology**: The OTN (***O**ntología de **T**ítulos u**N**iversitarios*[70]), a Universidad San Pablo CEU initiative, is intended to specify, in a formal way, a conceptualization for the exchange of information according to the Practical Guide for the Evaluation and Monitoring of Official Degrees of the Madri+d Foundation (Madri+d, 2018). It covers the information on official degrees that has to be public according to the Madri+d Foundation. This guide is applicable to all fourteen universities of Madrid.[71] The knowledge sources used during this project were: a) The Spanish series of Royal Decrees on Higher Education (BOE, 2018f,a,b,c,d,e); b) The aforementioned Madri+d guide; c) The guide for accreditation of degrees of the National Agency for Quality Assessment and Accreditation of Spain (ANECA) (ANECA, 2018); and d) An experienced coordinator of a degree in Information Systems.
- **Ciudades Abiertas**: The Spanish project Ciudades Abiertas,[72] (aka Open Cities), in which four town-halls are part of the project (A Coruña, Madrid, Santiago de Compostela and Zaragoza), aims at homogenizing the access to different cities APIs and Open Data portals by developing and sharing a set of ontologies. A total of twelve ontologies representing several domains related to cities are being developed: local business census and activity

[63] https://saref.etsi.org/core/.
[64] https://www.etsi.org/committee/smartm2m.
[65] https://portal.etsi.org/STF/STFs/STF-HomePages/STF534.
[66] https://www.delta-h2020.eu/.
[67] https://www.vicinity2020.eu.
[68] https://bimerr.eu.
[69] https://easytvproject.eu/.
[70] https://w3id.org/def/otn.
[71] http://www.emes.es/Sistemauniversitario/UniversidadesdeMadrid/tabid/215/Default.aspx.
[72] https://ciudadesabiertas.es/.

Table 2
Methodological technique (M) and Technology (T) used in the international projects where the framework has been followed.

Project	Team configuration	Project type	Ontology requirements specification		Ontology implementation			Ontology publication		Support and maintenance
			Knowledge sources	Requirements	Conceptualization	Encoding	Evaluation	Documentation	Online publication	Support along the whole project
SAREF4AGRI (09/2017–04/2019)	Ontology Engineer (1) access to DE consultancy	ETSI project (Standarization)	M: Interviews, data models inspection, use cases T: Google spreadsheet, physical meetings, online meetings, email	M: Use cases, CQ, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Data example coverage T: Reasoners, OOPS!	M: ETSI Technical Report and ETSI Technical specification, Human-readable description, ontology metadata, diagrams, examples T: Microsoft word, Widoco, diagrams.net	M: Under ETSI SAREF Ontology portal T: w3id through OnToology migrated to ETSI domain and servers	M: – T: GitHub (1 repository for all SAREF extensions), OnToology migrated to ETSI forge GitLab (1 repository per extension
SAREF4INMA (09/2017–04/2019)	Ontology Engineer, Domain Expert	ETSI project (Standarization)	M: Data models inspection, standards inspection, use cases T: Google spreadsheet, online meetings	M: Use cases, CQ, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Verification, Data example coverage T: Reasoners, OOPS!, Themis	M: ETSI Technical Report and ETSI Technical specification, Human-readable description, ontology metadata, diagrams, examples T: Microsoft word, Widoco, diagrams.net	M: Under ETSI SAREF Ontology portal T: w3id through OnToology migrated to ETSI domain and servers	M: – T: GitHub (1 repository for all SAREF extensions), OnToology migrated to ETSI forge GitLab (1 repository per extension)
SAREF4CITY (09/2017–04/2019)	Ontology Engineer	ETSI project (Standarization)	M: Data models inspection, standards inspections T: Google spreadsheets	M: Use cases, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Data example coverage T: Reasoners, OOPS!	M: ETSI Technical Report and ETSI Technical specification, Human-readable description, ontology metadata, diagrams, examples T: Microsoft word, Widoco, diagrams.net	M: Under ETSI SAREF Ontology portal T: w3id through OnToology migrated to ETSI domain and servers	M: – T: GitHub (1 repository for all SAREF extensions), OnToology migrated to ETSI forge GitLab (1 repository per extension)
SAREF4WATR (12/2018–05/2020)	Ontology Engineer, Domain Experts	ETSI project (Standarization)	M: Interviews with domain experts, data models inspection, standards inspection, workshops T: Text documents y online meetings	M: Use cases, CQ, statements T: Spreadsheet and text document	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege, text editor	M: Logical consistency checking, Validation, Verification, Data example coverage T: Reasoners, OOPS!, Themis	M: ETSI Technical Report and ETSI Technical specification, Human-readable description, ontology metadata, diagrams, examples T: Microsoft word, Widoco, diagrams.net	M: Under ETSI SAREF Ontology portal T: Publisher's domain and server	M: – T: ETSI forge GitLab (1 repository per extension)
DELTA (08/2018–04/2020)	Ontology Engineer , Domain Expert, Software Developer	European Project	M: Interviews, tables T: Google spreadsheet, online meetings	M: Ad.hoc tables with template, sentences T: Google spreadsheets	UML_Ont diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Verification, Data coverage T: Reasoners, OOPS!, Themis	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, diagrams.net	M: Ontology portal (2 ontologies) T: One ontology published under w3id through OnToology and one ontology published under publisher's domain and server	M: – T: GitHub (1 repository for each ontology), OnToology
VICINITY (01/2016–09/2017)	Ontology Engineer, Domain Expert, Software Developer	European Project	M: Documentation exchange, API descriptions, Data models inspection, standards inspections T: Google spreadsheet, physical meetings, online meetings, email	M: Ad.hoc tables with templates CQ, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: omnigraffle	M: – T: Protege	M: Logical consistency checking, Validation, Verification, Data coverage T: Reasoners, OOPS!, Themis	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, omnigraffle	M: Ontology portal (5 ontologies) T: Publisher's domain and server	M: – T: GitHub (1 repository for each ontology), OnToology
BIMERR (08/2019–06/2021)	Ontology Engineer, Domain Expert, Software Developer	European Project	M: Data model descriptions, standard specifications, meetings T: Confluence, physical meetings, online meetings, email	M: Tables templates custom based on METHONTOLOGY, glossary of terms T: Confluence	M: Chowlk Visual Notation T: diagrams.net	M: – T: Chowlk, Protege	M: Logical consistency checking, Validation. (Ongoing: verification and data example coverage) T: Reasoners, OOPS! (Ongoing: Themis)	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, diagrams.net	M: Ontology portal (9 ontologies) T: Publisher's domain and server	M: – T: GitHub (1 repository for each ontology), OnToology
easyTV (01/2017–06/2019)	Ontology Engineer, Domain Expert, Software Developer	European Project	M: Data model descriptions T:Google spreadsheet, physical meetings, online meetings, email	M: CQ, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: omnigraffle	M: – T: Protege	M: Logical consistency checking, Validation, Data coverage. Application based validation T: Reasoners, OOPS!, SPARQL queries	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, omnigraffle	M: Ontology published T: w3id through OnToology redirected to GitHub	M: – T: GitHub (1 repository), OnToology

Table 3
Methodological technique (M) and Technology (T) used in the regional and private company projects where the framework has been followed.

Project	Team configuration	Project type	Ontology requirements specification		Ontology implementation			Ontology publication		Support and maintenance
			Knowledge sources	Requirements	Conceptualization	Encoding	Evaluation	Documentation	Online publication	Support along the whole project
OTN (12/2017–07/2019)	Ontology Engineer, Domain expert (2) The Ontology Engineer has also the role of Domain Expert	Spanish regional project	M: Documents inspection T: LaTex	M: Glossary of terms T: LaTex	M: UML diagrams T: Umbrello	M: – T: Protege	M: Logical consistency checking, Validation T: Reasoners, OOPS!	M:Human-readable description T: Widoco	M: Ontology published T: w3id through OnToology	M: – T: GitHub (1 repository)
Ciudades Abiertas (07/2018–12/2020)	Ontology Engineer, Domain Expert	Spanish National project	M: Data inspection, data model inspection, Spanish laws inspection T: Online meetings with experts and data chiefs of city councils	M: Use cases, users stories, CQ, glossary of terms T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Data coverage T: Reasoners, OOPS!	M: Human-readable description, ontology metadata, diagrams, examples of use and example of SPARQL queries T: Microsoft word, Widoco, diagrams.net	M: Under Ciudades Abiertas portal T: Publisher’s domain and server	M: – T: GitHub (11 repositories)
BTN100 (11/2017–01/2018)	Ontology Engineer, Domain Expert	Private company project)	M: Data inspection, data model inspection T: Meetings with domain experts	M: CQ T: Google spreadsheet	M: UML_Ont-based diagrams T: Microsoft Visio	M: – T: Protege	M: Logical consistency checking, Validation, Data coverage T: Reasoners, OOPS!, Themis	M: Human-readable description, ontology metadata, diagrams, examples of use and example of SPARQL queries T: Microsoft word, Widoco, diagrams.net	M: Under datos.ign portal T: Publisher’s domain and server	M: – T: Github (1 repository)
Noise ontology (03/2017–06/2017)	Ontology Engineer, Domain Expert	Spanish National project	M: Data inspection, standards inspection T: meetings with domain experts	M: CQ T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege	M: Logical consistency checking, Validation, Data coverage T: Reasoners, OOPS!	M: Human-readable description, ontology metadata, diagrams, examples T: Microsoft word, Widoco, diagrams.net	M: Under vocab.linkeddata.es portal T:Publisher’s domain and server	M: – T: Github (1 repository)
SDH (03/2015–07/2016)	Ontology Engineer, Domain Expert, Software Developer	Private company project	M: APIs documentation T: Text files, annotated diagrams, physical meetings, emails	M: Sentences T: Text files, annotated diagrams	UML_Ont diagrams T: omnigraffle	M: – T: Protege	M: Logical consistency checking, Validation, Data coverage. Application based validation T: Reasoners, OOPS!, SPARQL queries	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, omnigraffle	M: No ontology publication according to LD principles T: –	M: – T: GitHub (1 repository with 15 ontologies)
MASS MEDIATOR (01/2019–04/2019)	Ontology Engineer, Domain Expert, Software Developer (3) The Ontology Engineer has also the role of Domain Expert and Software Developer	Spanish regional project	M: Specification of REST API T: –	M: – T: –	M: UML_Ont-based diagrams T: Umbrello	M: – T: Protege	M: Logical consistency checking T: Reasoners	M: Human-readable description, examples T: Widoco	M: Ontology published T: w3id through OnToology	M: – T: GitHub (1 repository)
Private company project 1 (07/2019–01/2020)	Ontology Engineer, Software Developer (1) access to DE consultancy	Private company project	M: Insurance product specification, interviews T: Google spreadsheet, physical meetings	M: Glossary of terms, sentences T: Google spreadsheet	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege, Google Refine	M: Logical consistency checking, Data coverage T: Reasoners, OOPS!, SPARQL queries	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, diagrams.net	M: private T: –	M: – T: Google drive migrated to company’s GitLab
Private company project 2 (05/2020–07/2020)	Ontology Engineer, Software Developer (1) access to DE consultancy	Private company project	M: Insurance product specification, emails for questions T: Google spreadsheet, online meetings	M: Glossary of terms T: Google spreadsheet	M: Chowlk Visual Notation T: diagrams.net	M: – T: Protege, Google Refine	M: Logical consistency checking, Data coverage T: Reasoners, OOPS!, SPARQL queries	M: Human-readable description, ontology metadata, diagrams, examples T: Widoco, diagrams.net	M: private T: –	M: – T: GitHub (1 private repository)
CASO & IRRIG (03/2018–01/2020)	Ontology Engineer, Software Developer, Domain Expert	French regional project	M: Irrigation method document, emails for questions, experimental dataset T: Pdf reader, online meetings (skype, retenater), email, excel, google sheet, google docs	M: Terms and Sentences, Competency Questions, Aggregation Formula T: Google docs, google spreadsheets	M: UML_Ont-based diagrams, State machine T: diagrams.net	M: – T: Protege, custom scripts, SWRL rules through Protege	M: Logical consistency checking, Data coverage, Validation, Unit tests, Sample tests, System tests T: OOPS!, Drools Engine, SWRL Tabs	M:Human-readable description, ontology metadata, diagrams, examples T: Widoco, diagrams.net	M: Ontology portal (2 ontologies) T: w3id through Ontoology	M: – T: GitHub (2 repositories), irstea GitLab for the examples, OnToology
D2KAB (6/2019–12/2023)	Ontology Engineer, Domain Expert	French regional project	M: French Wine and Vine Institute (IFV)’s French labels, BBCH (Biologische Bundesanstalt, Bundessortenamt und CHemische Industrie) monography and scientific papers T: Excel files, google slides, pdf reader	M: Competency Questions T: Google docs	M: UML_Ont-based diagrams T: diagrams.net	M: – T: Protege, Cellfie plugin, SWRL Tabs with Drowl reasoner	M: thesaurus consistency checking (label, description, schema, …), mappings between concepts T: SKOS play, Drowl reasoner	M: Human description for grapevine experts T: google slide	M: Agroportal T: GitLab URL, SPARQL endpoint	M: – T: Gitlab

licenses, acoustic pollution, census of inhabitants, events that take place within the municipal agenda and their participants, public bicycle system, traffic, agreements adopted by municipalities, public debt, public employment offer, budget information and budget execution, public transport by bus, and grants.

- **datos.ign.es**: Datos.ign.es[73] is an Instituto Geográfico Nacional (IGN)[74] initiative for the generation of semantic information from its resources. The first ontology resulting of this project is the BTN100 ontology,[75] which aims to represent the geospatial data from the Spanish Topographic Base Catalog (1:100.000 scale). The BTN100 catalog contains geographic information about topographic and thematic data. The BTN100 ontology allows representing all the complex geometrical shapes, e.g. multi-lines, polygons, multi-polygons, etc. specified in this catalog.
- **The Noise ontology**: The Noise ontology (Espinoza-Arias et al., 2019) aims to represent several noise levels, specific details about noise sensors, such as weighting methods or sensor classes, and different types of acoustic emitters. This ontology has been included as an example in the open data guide[76] published by the Spanish Federation of Municipalities and Provinces (FEMP), which aims to present a work itinerary on the opening of data and its reuse for all Spanish local administrations.
- **SDH**: The Smart Developer Hub project[77] had as purpose to integrate the data of different development and support tools during software building (Gitlab, Github, Jira, etc.). Such data would be queried by a dashboard to monitor the contributions in the different projects of the company. It was carried out within a collaboration frame between Banco de Santander and the Ontology Engineering Group (OEG). In this project the ontologies were not meant to be published online using content negotiation processes.
- **CEU Mass Mediator**: One of the main problems in biochemistry is, given a collection of empirical data about chemical compounds, to know to which compounds the data correspond. In the context of this problem, the Centre of Metabolomics and Bioanalysis and the Escuela Politécnica Superior (both of them belonging to Universidad San Pablo CEU) have developed CEU Mass Mediator.[78] It is a tool for database searching that allows the simultaneous batch query in different databases: METLIN,[79] KEGG,[80] Lipid Maps,[81] HMDB,[82] MINE[83] and an in-house library. Given that query results include multiple compound candidates for each empirical data, a Prolog expert system has been developed to aid in the identification of the right candidate (Fernández-López et al., 2019c). Such expert system has been encapsulated in an API REST service (fully written in Prolog as well). The specification of JSON objects has been carried out by means of an OWL ontology.[84]
- **Private company projects**: Two ontology development projects have been carried out (i.e., "Private company project 1" and "Private company project 2" in Table 3) where the ontology was built to annotate company customers data. For privacy and strategic issues the domain in which the ontologies were applied is not disclosed.[85]
- **The CASO & IRRIG ontology**: The Context-Aware System (CASO) and Irrigation (IRRIG) ontologies were developed in the context of a French national project to support the development of a smart irrigation context aware system. In this case the ontology development process was aligned with a mini-waterfall software development process (Nguyen et al., 2020).
- **D2KAB**. The French project *Data to Knowledge in Agronomy and Biodiversity* illustrates how semantic data science helps the development of innovative agricultural applications. The goal of D2KAB is to create a framework to turn agronomy and biodiversity data into semantically described, interoperable, actionable, and open knowledge. This project uses AgroPortal repository.[86] to find, publish and share semantic resources (Roussey et al., 2020)

[73] https://datos.ign.es/.

[74] http://www.ign.es/web/ign/portal.

[75] https://datos.ign.es/def/btn100.

[76] https://datos.gob.es/es/documentacion/datos-abiertos-guia-estrategica-para-su-puesta-en-marcha-y-conjuntos-de-datos-0.

[77] https://github.com/SmartDeveloperHub/sdh-vocabulary.git.

[78] http://ceumass.eps.uspceu.es/mediator/.

[79] https://metlin.scripps.edu/index.php.

[80] http://www.genome.jp/kegg/.

[81] http://www.lipidmaps.org/.

[82] http://www.hmdb.ca/.

[83] http://minedatabase.mcs.anl.gov/#/home.

[84] Both the API REST and the ontology are available at https://bitbucket.org/marianofl1971/mediatorkbservice/src/master/.

[85] This information could be provided to reviewers if requested in a confidential way.

[86] http://agroportal.lirmm.fr.

References

Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Perez, I., Fernández-Izquierdo, A., Corcho, O., 2019. Automating ontology engineering support activities with ontoology. J. Web Semant. 57, 100472.

ANECA, 2018. Guía de apoyo para la elaboración de la memoria de verificación de títulos oficiales universitarios (grado y máster). http://www.aneca.es/content/download/12155/136031/file/verifica_gm_guia_V05.pdf. (Accessed 20 July 2018).

Auer, S., 2006. RapidOWL—An agile knowledge engineering methodology. In: Proceedings Of 1st International Workshop On Semantic Technologies In Collaborative Applications (STICA 06), 26th-28th June, Manchester, UK. IEEE Computer Society (P2623).

Beck, K., 2000. Extreme Programming Explained: Embrace Change. Addison-wesley professional.

BOE, 2018a. Ley Orgánica 6/2001, de 21 de diciembre, de Universidades. http://www.boe.es/buscar/pdf/2001/BOE-A-2001-24515-consolidado.pdf. (Accessed 20 July 2018).

BOE, 2018b. Real Decreto 1125/2003, de 5 de septiembre, por el que se establece el sistema europeo de créditos y el sistema de calificaciones en las titulaciones universitarias de carácter oficial y validez en todo el territorio nacional. https://www.boe.es/boe/dias/2003/09/18/pdfs/A34355-34356.pdf. (Accessed 20 July 2018).

BOE, 2018c. Real Decreto 1393/2007, de 29 de octubre, por el que se establece la ordenación de las enseñanzas universitarias oficiales. https://www.boe.es/boe/dias/2007/10/30/pdfs/A44037-44048.pdf. (Accessed 20 July 2018).

BOE, 2018d. Real Decreto 1791/2010, de 30 de diciembre, por el que se aprueba el Estatuto del Estudiante Universitario. https://www.boe.es/boe/dias/2010/12/31/pdfs/BOE-A-2010-20147.pdf. (Accessed 20 July 2018).

BOE, 2018e. Real decreto 861/2010, de 2 de julio, por el que se modifica el real decreto 1393/2007, de 29 de octubre, por el que se establece la ordenación de las enseñanzas universitarias oficiales. https://www.boe.es/boe/dias/2010/07/03/pdfs/BOE-A-2010-10542.pdf. (Accessed 20 July 2018).

BOE, 2018f. Real Decreto 898/1985, de 30 de abril, sobre régimen del profesorado universitario. https://www.boe.es/buscar/pdf/1985/BOE-A-1985-11578-consolidado.pdf. (Accessed 20 July 2018).

Carriero, V.A., Daquino, M., Gangemi, A., Nuzzolese, A.G., Peroni, S., Presutti, V., Tomasi, F., 2020. The Landscape of Ontology Reuse Approaches. In: Applications and Practices in Ontology Design, Extraction, and Reasoning, IOS Press, pp. 21–38.

Chávez-Feria, S., García-Castro, R., Poveda-Villalón, M., 2021. Converting UML-based ontology conceptualizations to OWL with chowlk. In: Verborgh, R., Dimou, A., Hogan, A., d'Amato, C., Tiddi, I., Bröring, A., Mayer, S., Ongenae, F., Tommasini, R., Alam, M. (Eds.), The Semantic Web: ESWC 2021 Satellite Events. Springer International Publishing, pp. 44–48.

Chen, P.P.-S., 1976. The entity-relationship model—toward a unified view of data. ACM Transactions On Database Systems 1 (1), 9–36. http://dx.doi.org/10.1145/320434.320440.

Codescu, M., Kuksa, E., Kutz, O., Mossakowski, T., Neuhaus, F., 2017. Ontohub: A semantic repository engine for heterogeneous ontologies. Appl. Ontology 12 (3–4), 275–298.

d'Aquin, M., 2012. Modularizing Ontologies. In: Ontology Engineering In A Networked World, pp. 213–233.

d'Aquin, M., Noy, N., 2012. Where to publish and find ontologies? A survey of ontology libraries. Web Semant. Sci. Serv. Agents World Wide Web 11, 96–111.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M., 2007. Ontology modularization for knowledge selection: Experiments and evaluations. In: Proceedings Of The 18th International Conference On Database And Expert Systems Applications, DEXA 2007, Regensburg, Germany, September 3-7, 2007. Springer, pp. 874–883.

Davies, K., Keet, C.M., Lawrynowicz, A., 2019. More effective ontology authoring with test-driven development and the TDDonto2 tool. Int. J. Artif. Intell. Tools 28 (07), 1950023.

de Hoog, R., 1998. Methodologies for building knowledge based systems: achievements and prospects. In: Liebowitz, J. (Ed.), Handbook Of Expert Systems. CRC Press, Boca Raton, Florida, pp. 1–14.

De Nicola, A., Missikoff, M., 2016. A lightweight methodology for rapid ontology engineering. Commun. ACM 59 (3), 79–86.

de Roode, M., Fernández-Izquierdo, A., Daniele, L., Poveda-Villalón, M., García-Castro, R., 2020. SAREF4INMA: A SAREF Extension for the industry and manufacturing domain. Semant. Web (Preprint), 1–16.

Espinoza-Arias, P., Corcho, O., 2018. Guías rápidas para la creación y uso de vocabularios consensuados para publicar datos abiertos. 3. šcómo desarrollar vocabularios?. http://dx.doi.org/10.5281/zenodo.1410871.

Espinoza-Arias, P., Poveda-Villalón, M., Corcho, O., 2019. Using LOT methodology to develop a noise pollution ontology: a spanish use case. J. Ambient Intell. Humaniz. Comput..

ETSI, 2020. ETSI TS 103 673 V1.1.1. SmartM2M; SAREF Development Framework and Workflow, Streamlining the Development of SAREF and its Extensions, ETSI. Technical Report.

Fernández-Izquierdo, A., García-Castro, R., 2018. Requirements behaviour analysis for ontology testing. In: Proceedings Of The 21st International Conference On Knowledge Engineering And Knowledge Management, EKAW 2018, Nancy, France, November 12-16, 2018. Springer, pp. 114–130.

Fernández-Izquierdo, A., García-Castro, R., 2019a. Themis: a tool for validating ontologies through requirements. In: Proceedings Of The International Conference On Software Engineering And Knowledge Engineering, SEKE 2019, Lisbon, Portugal, July 10-12, 2019. KSI Research Inc, pp. 573–578.

Fernández-Izquierdo, A., Poveda-Villalón, M., García-Castro, R., 2019b. CORAL: a corpus of ontological requirements annotated with lexico-syntactic patterns. In: Proceedings Of The 16th Extended Semantic Web Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019. Springer, pp. 443–458.

Fernández-López, M., Gil de la Fuente, A., Godzien, J., Rupérez, J., Barbas, C., Otero, A., 2019c. LAS: A lipid annotation service with explanations. Comput. Struct. Biotechnol. J. (Under the second round of revision).

Fernández-López, M., Gómez-Pérez, A., Juristo, N., 1997. METHONTOLOGY: From ontological art towards ontological engineering. In: Proceedings Of The Ontological Engineering AAAI97 Spring Symposium Series. American Asociation for Artificial Intelligence.

Fernández-López, M., a Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A., 2019. Why are ontologies not reused across the same domain? J. Web Semant. 57, 100492.

Fernández-López, M., Suárez-Figueroa, M.C., Gómez-Pérez, A., 2012. Ontology development by reuse. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (Eds.), Ontology Engineering In A Networked World. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 147–170.

Gangemi, A., Presutti, V., 2009. Ontology design patterns. In: Handbook On Ontologies. Springer, pp. 221–243.

García-Castro, R., Fernández-Izquierdo, A., Heinz, C., Kostelnik, P., Poveda-Villalón, M., Serena, F., 2017. D2.2 detailed specification of the semantic model. Universidad Politécnica de Madrid (UPM), URL https://vicinity2020.eu/vicinity/node/229.

Garijo, D., 2017. WIDOCO: A Wizard for documenting ontologies. In: Proceedings Of The 16th International Semantic Web Conference, ISWC 2017, Vienna, Austria, October 21-25, 2017. Springer International Publishing, Cham, pp. 94–102.

Garijo, D., Poveda-Villalón, M., 2020. Best practices for implementing FAIR vocabularies and ontologies on the web. In: Applications and Practices in Ontology Design, Extraction, and Reasoning, IOS Press, pp. 39–54.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5 (2), 199–220.

Grüninger, M., Fox, M.S., 1995. Methodology for the Design and Evaluation of Ontologies. In: Proceedings Of The Workshop On Basic Ontological Issues In Knowledge Sharing, Held In Conjunction With IJCAI-95.

Gyrard, A., Bonnet, C., Boudaoud, K., Serrano, M., 2016. LOV4IoT: A Second life for ontology-based domain knowledge to build semantic web of things applications. In: Proceedings Of The 4th IEEE International Conference On Future Internet Of Things And Cloud, FiCloud 2016, Vienna, Austria, August 22-24, 2016. IEEE, pp. 254–261.

Haase, P., Brockmans, S., Palma, R., Euzenat, J., d'Aquin, M., 2009. D1.1.2 updated version of the networked ontology model. NeOn Project Deliverable URL http://neon-project.org/deliverables/WP1/NeOn_2007_D1.1.2.pdf.

Halilaj, L., Petersen, N., Grangel-González, I., Lange, C., Auer, S., Coskun, G., Lohmann, S., 2016. Vocol: An integrated environment to support version-controlled vocabulary development. In: Proceedings Of The 20th International Conference On Knowledge Engineering And Knowledge Management, EKAW 2016, Bologna, Italy, November 19-23, 2016. Springer, pp. 303–319.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., 2012. OWL 2 web ontology language primer. W3C Recomm. http://www.w3.org/TR/owl2-overview/.

Horridge, M., Gon̦ calves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A., 2019. WebProtéGé: A Cloud-based ontology editor. In: Companion Proceedings Of The 2019 World Wide Web Conference, San Francisco, CA, USA, May 13-17, 2019. Association for Computing Machinery, pp. 686–689.

Hristozova, M., Sterling, L., 2002. An extreme method for developing lightweight ontologies. In: Proceedings Of The Workshop On Ontologies In Agent Systems, 1st International Joint Conference On Autonomous Agents And Multi-Agent Systems, Bologna, Italy, 16 July 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.3690.

Hugo, W., Le Franc, Y., Coen, G., Parland-von Essen, J., Bonino, L., 2020. D2. 5 FAIR semantics recommendations second iteration. http://dx.doi.org/10.5281/zenodo.4314321.

IEEE, IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998, pp. 1–40. http://dx.doi.org/10.1109/IEEESTD.1998.88286.

Jonquet, C., Toulet, A., Arnaud, E., Aubin, S., Yeumo, E.D., Emonet, V., Graybeal, J., Laporte, M.-A., Musen, M.A., Pesce, V., et al., 2018. AgroPortal: A Vocabulary and ontology repository for agronomy. Comput. Electron. Agric. 144, 126–143.

Käbisch, S., Kamiya, T., McCool, M., Charpenay, V., Kovatsch, M., 2020. Web of things (WoT) thing description. W3C recommendation. W3C. In: W3C Recommendation. World Wide Web Consortium (W3C), https://www.w3.org/TR/wot-thing-description/.

Kamruzzaman, S.M., Krisnadhi, A., Hitzler, P., 2016. OWLAx: a protégé plugin to support ontology axiomatization through diagramming. In: Proceedings Of The ISWC 2016 Posters & Demonstrations Track Co-Located With 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016.

Katsumi, M., Grüninger, M., 2016. What Is Ontology Reuse? In: Proceedings Of The 9th International Conference On Formal Ontology In Information Systems, FOIS 2016, Annecy, France, July 6-9, 2016, pp. 9–22.

Knublauch, H., Kontokostas, D., 2017. Shapes constraint language (SHACL). W3C Recomm. URL https://www.w3.org/TR/shacl/.

Kotis, K.I., Vouros, G.A., Spiliotopoulos, D., 2020. Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations. Knowl. Eng. Rev. 35, e4.

Lohmann, S., Link, V., Marbach, E., Negru, S., 2014. WebVOWL: WEb-based visualization of ontologies. In: Proceedings Of The International Conference On Knowledge Engineering And Knowledge Management, EKAW 2014 Satellite Events, VISUAL, EKM1, And ARCOE-Logic, LinkÖPing, Sweden, November 24-28, 2014. Springer, pp. 154–158.

Madri+d, F., 2018. Guía Práctica para la Evaluación del Seguimiento de Títulos Universitarios Oficiales de la Fundación Madri+d. http://www.madrimasd.org/uploads/acreditacion/seguimiento2013/Guia_de_Evaluacion_2014.pdf. (Accessed 20 July 2018).

Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., Parkinson, H., 2010. Modeling sample variables with an experimental factor ontology. Bioinformatics 26 (8), 1112–1118.

McDaniel, M., Storey, V.C., 2019. Evaluating domain ontologies: clarification, classification, and challenges. ACM Comput. Surv. 52 (4), 1–44.

Miles, A., Bechhofer, S., 2009. SKOS Simple knowledge organization system reference. W3C Recommendation.

Musen, M.A., 2015. The protégé project: a look back and a look forward. AI Matters 1 (4), 4–12. http://dx.doi.org/10.1145/2757001.2757003.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., Swartout, W.R., 1991. Enabling technology for knowledge sharing. AI Mag. 12 (3), 36–56.

Negru, S., Lohmann, S., 2013. A visual notation for the integrated representation of OWL ontologies. In: Proceedings Of The 9th International Conference On Web Information Systems And Technologies, Aachen, Germany, 8-10 May,2013. SciTePress, pp. 308–315.

Nguyen, Q.-D., Roussey, C., Poveda-Villalón, M., de Vaulx, C., Chanet, J.-P., 2020. Development experience of a context-aware system for smart irrigation using CASO and IRRIG ontologies. Appl. Sci. 10 (5).

Noy, N.F., McGuinness, D.L., 2001. Ontology development 101: A guide to creating your first ontology. URL http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html.

Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.-A., Chute, C.G., et al., 2009. BioPortal: Ontologies and integrated data resources at the click of a mouse. Nucleic Acids Res. 37, 170–173.

Ong, E., Xiang, Z., Zhao, B., Liu, Y., Lin, Y., Zheng, J., Mungall, C., Courtot, M., Ruttenberg, A., He, Y., 2017. Ontobee: a linked ontology data server to support ontology term dereferencing, linkage, query and integration. Nucleic Acids Res. 45 (D1), D347–D352.

Peroni, S., 2016. A simplified agile methodology for ontology development. In: Proceedings Of The 13th International Workshop On OWL: - Experiences And Directions - Reasoner Evaluation, OWLED 2016, And 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016. Springer, pp. 55–69.

Peroni, S., Shotton, D., Vitali, F., 2012. The live OWL documentation environment: a tool for the automatic generation of ontology documentation. In: Proceedings Of The 18th International Conference On Knowledge Engineering And Knowledge Management, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Springer, pp. 398–412.

Pinto, H.S., Staab, S., Tempich, C., 2004. DILIGENT: TOwards a fine-grained methodology for DIstributed, loosely-controlled and evolvInG Engineering of oNTologies. In: Proceedings Of The 16th European Conference On Artificial Intelligence, ECAI'2004, Including Prestigious Applicants Of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004. pp. 393–397.

Pinto, H., Tempich, C., Staab, S., 2009. Ontology engineering and evolution in a distributed world using DILIGENT. In: Staab, S., Studer, R. (Eds.), Handbook On Ontologies. In: International Handbooks on Information Systems, Springer Berlin Heidelberg, pp. 153–176.

Poveda-Villalón, M., 2012. A reuse-based lightweight method for developing linked data ontologies and vocabularies. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (Eds.), Proceedings Of The 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 833–837.

Poveda-Villalón, M., Espinoza-Arias, P., Garijo, D., Corcho, O., 2020. Coming to terms with FAIR ontologies: A position paper. In: Proceedings Of The 22nd International Conference On Knowledge Engineering And Knowledge Management.

Poveda-Villalón, M.a., Gómez-Pérez, A., Suárez-Figueroa, M.C., 2014. OOPS! (OntOlogy Pitfall Scanner!): An On-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. 10 (2), 7–34.

Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A., 2012. The Landscape of Ontology Reuse in Linked Data. In: Proceedings Of Ontology Engineering In A Data-Driven World, OEDW 2012, Galway, Irland October 9, 2012.

Presutti, V., Blomqvist, E., Daga, E., Gangemi, A., 2012. Pattern-based ontology design. In: Ontology Engineering In A Networked World. Springer, pp. 35–64.

Presutti, V., Daga, E., Gangemi, A., Blomqvist, E., 2009. EXtreme Design with content ontology design patterns. In: Proceedings Of The Workshop On Ontology Patterns, Collocated With The 8th International Semantic Web Conference, Washington D.C., USA, 25 October, 2009. CEUR-WS.org, Aachen, DEU, pp. 83–97.

Roussey, C., Delpuech, X., Amardeilh, F., Bernard, S., Jonquet, C., 2020. Semantic Description of Plant Phenological Development Stages, starting with Grapevine. In: Proceedings Of The 14th International Conference On Metadata And Semantics Research (MTSR2020), Virtual Conference, 2-4 December, 2020.

Sabou, M., Fernandez, M., 2012. Ontology (network) evaluation. In: Ontology Engineering In A Networked World. Springer, pp. 193–212.

Schandl, T., Blumauer, A., 2010. PoolParty: SKOS Thesaurus management utilizing linked data. In: Proceedings Of The 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010. Springer, pp. 421–425.

Schober, D., Tudose, I., Svatek, V., Boeker, M., 2012. OntoCheck: Verifying ontology naming conventions and metadata completeness in protégé 4. In: Journal Of Biomedical Semantics. 3, (S2), Springer, p. S4.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y., 2007. Pellet: A practical owl-dl reasoner. Journal Of Web Semantics 5 (2), 51–53.

Staab, S., Studer, R., Schnurr, H.-P., Sure, Y., 2001. Knowledge processes and ontologies. IEEE Intell. Syst. 16 (1), 26–34.

Studer, R., Benjamins, V.R., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data Knowl. Eng. 25 (1–2), 161–197.

Suárez-Figueroa, M.C., Cea, G.A.d., Gómez-Pérez, A., 2013. Lights and shadows in creating a glossary about ontology engineering. Terminology 19 (2), 202–236.

Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M., 2015. The neon methodology framework: A scenario-based methodology for ontology development. Applied Ontology 10 (2), 107–145.

Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A., 2012. Introduction: ontology engineering in a networked world. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (Eds.), Ontology Engineering In A Networked World. Springer Verlag, Berlin Heidelberg, Germany, pp. 1–6.

Suárez-Figueroa, M.C., Gómez-Pérez, A., Villazón-Terrazas, B., 2009. How to write and use the ontology requirements specification document. In: Proceedings Of The OTM Confederated International Conferences" On The Move To Meaningful Internet Systems", Vilamoura, Portugal, November 1-6, 2009. Springer, pp. 966–982.

Tudorache, T., 2020. Ontology engineering: Current state, challenges, and future directions. Semantic Web 11 (1), 125–138. http://dx.doi.org/10.3233/SW-190382.

Vandenbussche, P.-Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B., 2017. Linked open vocabularies (LOV): a gateway to reusable semantic vocabularies on the web. Semantic Web 8 (3), 437–452.

Varzi, A., 2003. Mereology. Stanford Encyclopedia Of Philosophy URL https://plato.stanford.edu/entries/mereology/.

Villazón-Terrazas, B.M., 2012. A Method For Reusing And Re-Engineering Non-Ontological Resources For Building Ontologies. vol. 12, IOS Press.

Welty, C.A., Ferrucci, D.A., 1999. Instances and classes in software engineering. Intelligence 10 (2), 24–28.