



COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu

D3.3 – COGITO
Data Model and
Ontology
Definition and
Interoperability
Design v2



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 955310

D3.3 – COGITO Data Model and Ontology Definition and Interoperability Design v2

Dissemination Level:	Public
Deliverable Type:	Report
Lead Partner:	UPM
Contributing Partners:	Hypertech, UCL, AU, UEDIN, CERTH, BOC-AG, QUE, NT
Due date:	30-04-2022
Actual submission date:	29-04-2022

Authors

Name	Beneficiary	Email
Socorro Bernardos	UPM	sbernardos@fi.upm.es
María Poveda-Villalón	UPM	mpoveda@fi.upm.es
Raúl García-Castro	UPM	rgarcia@fi.upm.es
Giorgos Giannakis	Hypertech	g.giannakis@hypertech.gr
Georgios Lilis	UCL	g.lilis@ucl.ac.uk
Jochen Teizer	AU	teizer@eng.au.dk
Frédéric Bosché	UEDIN	f.bosche@ed.ac.uk
Vasilios Karkanis	CERTH	vkarkanis@iti.gr
Damiano Falcioni	BOC-AG	damiano.falcioni@boc-eu.com
Panagiotis Moraitis	QUE	p.moraitis@que-tech.com

Reviewers

Name	Beneficiary	Email
Agnieszka Mikołajczyk	ASM	a.mikolajczyk@asmresearch.pl
Elias Bruno Meusburger	RSRG	elias.meusburger@rsrg.com

Version History

Version	Editors	Date	Comment
0.1	Socorro Bernardos	15.03.2022	Table of Contents ready
0.4	Socorro Bernardos	07.04.2022	Sections 1-4
0.5	Socorro Bernardos	17.04.2022	Sections 5-6
0.6	Socorro Bernardos	19.04.2022	Complete draft version
0.7	A. Mikołajczyk, E. Meusburger	26.04.2022	Draft version peer reviewed
0.9	Socorro Bernardos	28.04.2022	Review comments addressed
1.0	Giorgos Giannakis, Socorro Bernardos	29.04.2022	Submission to the EC

Disclaimer

©COGITO Consortium Partners. All rights reserved. COGITO is a HORIZON2020 Project supported by the European Commission under Grant Agreement No. 958310. The document is proprietary of the COGITO consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights. The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies. The information and views presented in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use, which may be made, of the information contained therein.

Executive Summary

The COGITO ecosystem aims to enable interoperability with existing and emerging standards and data models covering different fields within the AEC domain. Semantic interoperability in the COGITO platform is based on ontologies that will be exploited throughout the COGITO infrastructure and used to semantically represent the information exchanged.

This document details the methodology and technological infrastructure used to develop the COGITO ontology network, as well as the second version of the ontologies that make up such an ontology network.

The followed methodology is called LOT [1] and includes four activities: 1) ontology requirements specification, 2) ontology implementation, 3) ontology publication, and 3) ontology maintenance.

The current version of the COGITO ontology network consists of four modules corresponding to the facility¹ (the construction itself), the construction process, the construction resources, and the quality achieved (see Table 3). These modules have already been conceptualised, implemented, and published and will be modified and complemented with new modules as the project progresses.

The next steps will be devoted to continuing the collection of more ontological requirements from the pilots as well as from the architecture to iteratively evolve the COGITO ontology network. The COGITO ontology portal will serve as a living repository for the different ontologies and will contain the latest version of the developed ontologies and all related artifacts.

Note here that since the methodology and infrastructure followed are the same as those described in D3.2 [2], this deliverable differs from that mainly in the sections that describe new developments in the ontology, namely Sections 4 and 5. It should be noted that in D3.2 [2] a coverage analysis was carried out to identify to what extent terms appearing in the requirements also appeared in the ontologies described in the previous D3.1 deliverable [3]. The conclusions highlighted the fact that reusing ontologies requires analysing multiple factors beyond coverage, and some of them are not easily quantifiable or even cannot be measured objectively. Based on this rationale, a coverage analysis for the new ontology requirements was not performed and therefore this particular section was removed.

¹ Please, have in mind that this module was called “construction”, instead of “facility”

Table of Contents

Executive Summary	3
Table of Contents	4
List of Figures	6
List of Tables	7
List of Acronyms and Abbreviations	8
1 Introduction	9
1.1 Scope and Objectives of the Deliverable	9
1.2 Relation to other Tasks and Deliverables	9
1.3 Updates to the first version of COGITO Data Model and Ontology Definition and Interoperability Design 9	
1.4 Structure of the Deliverable	10
2 Ontology Development Methodology	11
2.1 Ontological Requirements Specification	11
2.1.1 Identification of Purpose and Scope	12
2.1.2 Data Exchange Identification and Use Case Specification	12
2.1.3 Ontological Requirement Proposal	12
2.1.4 Completion and Validation of Ontology Requirements	13
2.1.5 Prioritisation of Ontological Requirements	13
2.2 Ontology Implementation	13
2.2.1 Ontology Conceptualisation	13
2.2.2 Ontology Encoding	14
2.2.3 Ontology Evaluation	14
2.3 Ontology Publication	14
2.3.1 Propose Release Candidate	15
2.3.2 Ontology Documentation	15
2.3.3 Online Publication	15
2.4 Ontology Maintenance	15
2.4.1 Bug Detection	16
2.4.2 New Requirements Proposal	16
3 Ontology Requirements Specification	17
4 Overview of the COGITO Ontology Network	19
4.1 Facility Module	20
4.2 Process Module	22
4.3 Resource Module	22
4.4 Quality Module	23
5 Ontology Development Infrastructure	25
5.1 Infrastructure to Support the Requirements Specification Activity	25

5.2	Infrastructure to Support the Ontology Implementation Activity	26
5.3	Infrastructure to Support the Ontology Publication Activity	26
5.3.1	Ontologies	26
5.3.2	How We Work	27
5.4	Infrastructure to Support the Ontology Maintenance Activity	27
6	Conclusions.....	29
	References.....	30

List of Figures

Figure 1: Ontology development methodology followed in COGITO	11
Figure 2: Workflow for the specification of the ontology requirements	12
Figure 3: Workflow for the ontology implementation activity	13
Figure 4: Workflow for the ontology publication activity	14
Figure 5: Workflow for the ontology maintenance activity	15
Figure 6: Overview of the COGITO ontology network	19
Figure 7: General overview of the COGITO facility ontology	21
Figure 8: General overview of the COGITO process ontology	22
Figure 9: General overview of the COGITO resource ontology	23
Figure 10: General overview of the COGITO quality ontology	24
Figure 11: COGITO ontology requirements for PMS	25
Figure 12: HTML requirements for the construction process	25
Figure 13: Ontology section in the COGITO ontology portal	27
Figure 14: Snapshot of the GitHub issue tracker	28

List of Tables

Table 1: General information on the specification of ontology requirements	17
Table 2: List of requirements per domain	17
Table 3: List of the ontologies created in the COGITO project.....	19
Table 4: List of the ontologies reused in the COGITO project	19

List of Acronyms and Abbreviations

Term	Description
AEC	Architecture Engineering & Construction
BBO	BPMN Based Ontology
BEO	Building Element Ontology
BOT	Building Topology Ontology
BPMN	Business Process Modelling Notation
BPO	Building Product Ontology
DICO	Digital Construction
ETSI	European Telecommunications Standards Institute
IoT	Internet of Things
LOT	Linked Open Terms
ORSD	Ontology Requirements Specification Document
OWL	Web Ontology Language
PMS	Process Modelling and Simulation
SAREF	Smart Applications REference ontology
SAREF4BLDG	SAREF extension for the Building domain
SSN	Semantic Sensor Network
SOSA	Sensor, Observation, Sample and Actuator
WODM	Work Order Definition and Monitoring tool

1 Introduction

1.1 Scope and Objectives of the Deliverable

The COGITO ecosystem aims to enable interoperability with existing and emerging standards and data models covering different fields in the AEC domain. Semantic interoperability in the COGITO platform is based on ontologies that will be exploited throughout the COGITO infrastructure and used to semantically represent the information exchanged.

In computer science, ontologies are defined as “formal, explicit specifications of a shared conceptualisation”. The COGITO ontologies will be developed using the W3C Web Ontology Language standard (OWL)² reusing existing resources and standards whenever possible. This development is based on the technical specification of the COGITO architecture and is based on a set of requirements extracted from its different components.

This deliverable describes the environment that supports the development of the COGITO ontologies. To do so, on the one hand, it describes the methodology to be followed to develop the ontologies and the ontology development infrastructure deployed to support such development. On the other hand, it presents the outcomes of the different ontology development sprints. It summarises all the requirements that were extracted from the COGITO use cases and architecture, which are used to analyse the coverage of existing ontologies to the project requirements. Finally, this document presents an overview of the current version of the COGITO ontologies that are available online on the COGITO ontology portal.³

The version of the ontology presented in this document is the version produced on the date of writing of this document. One version of this deliverable will be delivered in month 24 (October 2022) and will present the updated versions of the ontologies. In any case, the COGITO ontology portal will contain the latest version of the ontologies and all related artifacts for ontology development.

1.2 Relation to other Tasks and Deliverables

This deliverable is based on the use cases defined in D2.1 [4] and on the COGITO architecture defined in D2.4 [5]. Furthermore, D3.1 [3] was also considered to identify existing ontologies to reuse in the COGITO ontologies.

The COGITO ontology network described here will allow for sharing and interoperability among the different components of the COGITO architecture, especially regarding the communications with the Digital Twin platform. Hence, this document as well as future versions of the ontology will be particularly important during the development of each component comprising the COGITO architecture, i.e., the future results of work packages 4, 5, 6 and 7 and their integration in work package 8.

1.3 Updates to the first version of COGITO Data Model and Ontology Definition and Interoperability Design

This deliverable is the second one devoted to the development of the COGITO ontologies. Sections related to the ontology development methodology and the development infrastructure are identical (sections 2 and 5, respectively). The main changes compared to the previous version of the deliverable [2] are the following:

- The introduction and conclusions of the document have been updated.
- Section 3 has been updated including the new requirements gathered for the COGITO ontology since the previous version of the deliverable.
- Section 4 has been updated to include the current description of the ontology that includes the new requirements.
- The section related to the coverage analysis has been removed from the document. The conclusions previously drawn from this analysis highlighted the fact that reusing ontologies requires analysing

² <https://www.w3.org/TR/owl2-primer/>

³ <https://cogito.iot.linkeddata.es/>

multiple factors beyond coverage, and some of them are not easily quantifiable or cannot even be measured objectively.

1.4 Structure of the Deliverable

- **Section 2** provides an overview of the methodology used to develop the COGITO ontologies;
- **Section 3** presents the requirements to be satisfied by the COGITO ontologies;
- **Section 4** provides a description of the COGITO ontologies;
- **Section 5** presents the deployed ontology development infrastructure; and
- **Section 6** provides drawn conclusions and insights into future work.

2 Ontology Development Methodology

This section presents the ontology development methodology used for the development of the COGITO ontology network. This methodology includes four activities: 1) ontology requirements specification, 2) ontology implementation, 3) ontology publication, and 3) ontology maintenance. This development methodology is named Linked Open Terms (LOT) [1] and is based on the NeOn methodology [6]. It has been used and refined in previous ontology development processes from the European projects VICINITY [7], BIMERR [8] and DELTA [9], and adapted to the particularities of COGITO (see Sections 3 and 4).

Figure 1 shows an overview of the activities performed and the artefacts resulting from them: the ontology requirements specification document (ORSD), the ontology implementation, the ontology publication, and the ontology maintenance. The following subsections detail each activity (and resulting product). In the figures included in this section, these artefacts are represented in green squares and are assigned a number; the different activities in the methodology enumerate the numbers of all the input artefacts used in the activity in a green circle.

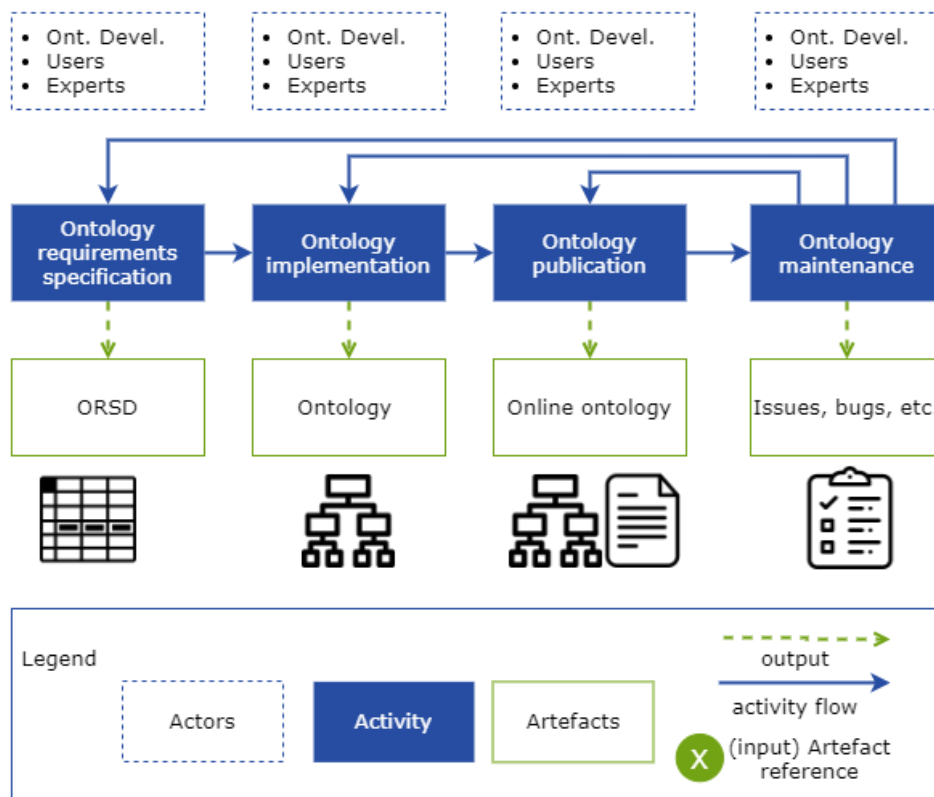


Figure 1: Ontology development methodology followed in COGITO

2.1 Ontological Requirements Specification

The aim of the requirements specification activity is to identify and define the requirements that each ontology to be created needs to fulfil. During this first activity, it is necessary to involve experts in the domain to capture the relevant industry perspective and knowledge. Figure 2 shows the workflow for the ontology requirements specification activity.

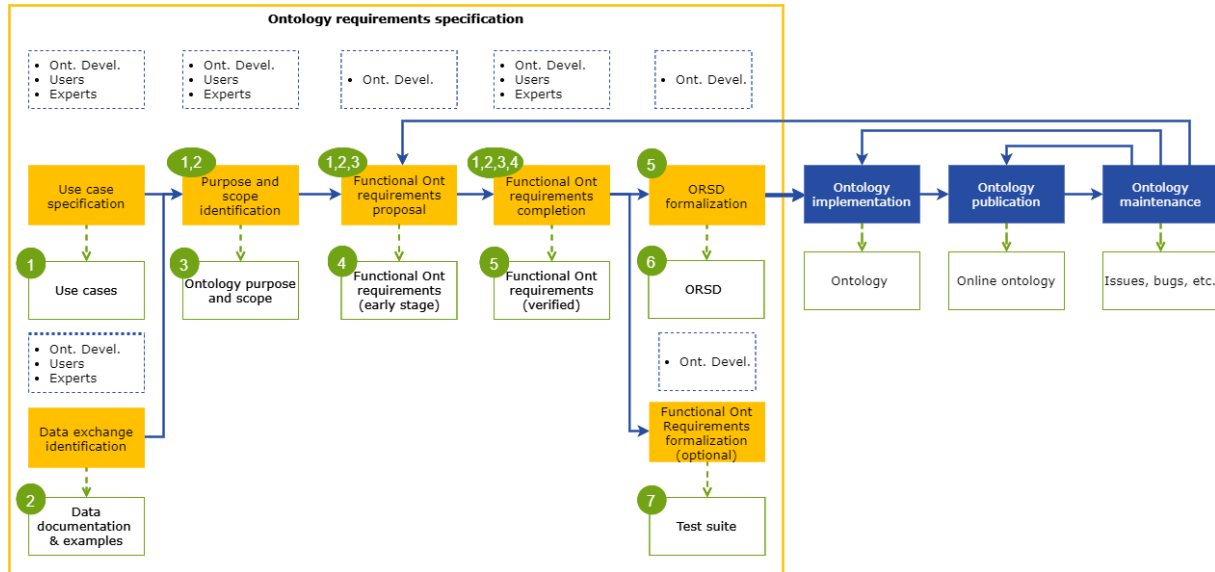


Figure 2: Workflow for the specification of the ontology requirements

2.1.1 Identification of Purpose and Scope

The goal of this step is to define the purpose and scope of a given ontology or module. To this end, the ontology development team works in collaboration with end-users and domain experts to define the purpose and scope of each ontology or module to be developed.

2.1.2 Data Exchange Identification and Use Case Specification

During this step, the ontology development team needs to gather the necessary documentation about the domain to be modelled. This documentation might correspond to the following:

- Standards
- Datasets
- API specifications
- Use cases

This documentation needs to be collected by both the ontology development team and the domain experts.

2.1.3 Ontological Requirement Proposal

Taking as input the documentation and the data provided by the domain experts and the end-users, the ontology development team generates a first proposal of the ontological requirements written in the form of competency questions or assertions.

The format used for this requirement proposal follows a tabular approach and includes the following fields:

- a requirement identifier, which must be unique for each requirement;
- the partner who proposed the requirement;
- the component of the COGITO architecture from which the requirement was extracted;
- the sprint in which the requirement is planned to be implemented;
- Competency questions or assertions;
- the status of the requirement, which can be: (1) Proposed, (2) Accepted, (3) Rejected, (4) Pending or (5) Deprecated;
- in case the requirement is deprecated, the identifier of the updated requirement is used;
- comments on the requirement;
- the provenance of the requirement, e.g., use case or standard;
- the priority of the requirement, which can be: (1) High, (2) Medium, or (3) Low.

The ontological requirements are completed iteratively, as the following ontology development process is incremental.

2.1.4 Completion and Validation of Ontology Requirements

During this activity, domain experts and end-users in collaboration with the ontology development team validate whether the ontology requirements defined in the previous step are correct and complete.

2.1.5 Prioritisation of Ontological Requirements

The prioritisation of the requirements allows teams to schedule the development of the ontology in sprints. In the COGITO project, this prioritisation will be performed if there is a need to prioritise functional requirements, mainly for those sprints with a high number of requirements in which the implementation of some of the requirements has to be postponed for a future sprint.

If prioritisation is needed, in order to execute it, the ontology development team works with the domain experts to identify which requirements need to be fulfilled first.

2.2 Ontology Implementation

During the implementation activity, the ontology is built using a formal language based on the requirements identified in the previous activity.

Taking the set of requirements collected in the previous activity as input, the ontology implementation activity is carried out through several sprints. To this end, the ontology developers schedule the ontology development according to the requirements that were identified, and the ontology development team builds the ontology iteratively by implementing only a certain number of requirements in each iteration. The output of each iteration is a new version of the ontology implementation. Figure 3 shows the steps that are followed in this ontology implementation activity.

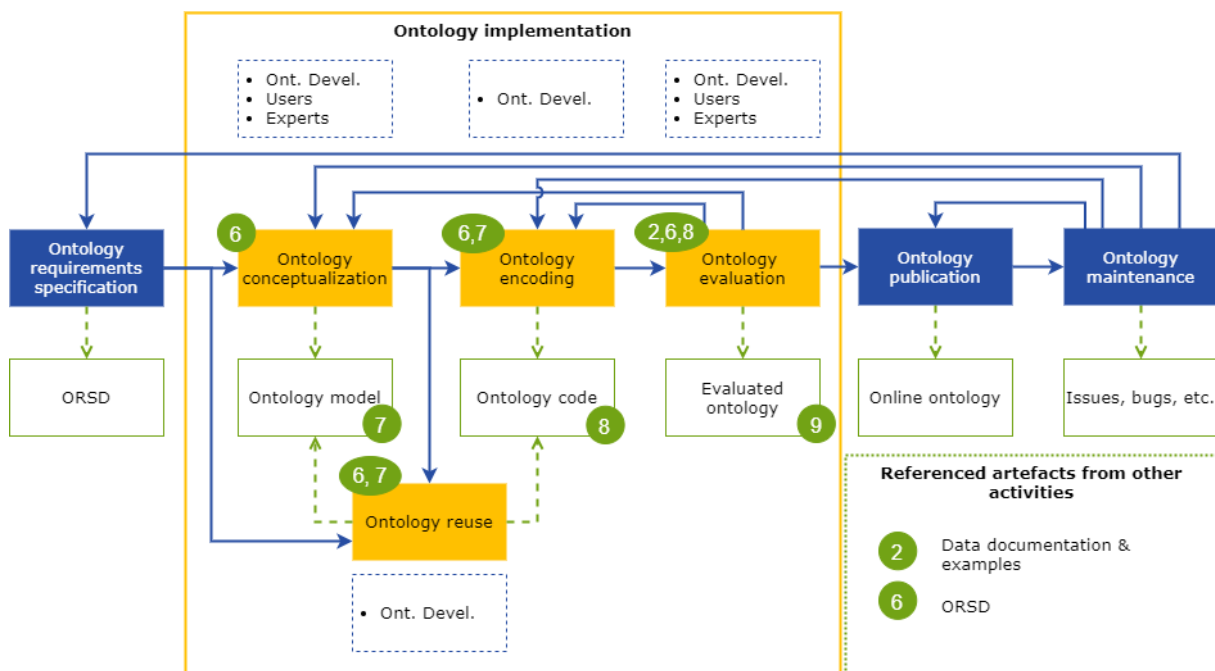


Figure 3: Workflow for the ontology implementation activity

2.2.1 Ontology Conceptualisation

The aim of this activity is to build an ontology model from the ontological requirements identified in the requirements specification process that represents the domain of the ontology. Therefore, during the

conceptualisation of the ontology, the domain knowledge obtained from the previous activity is organised and structured into a model by the ontology development team.

2.2.2 Ontology Encoding

The purpose of the encoding activity is to implement the ontology in an implementation language, such as OWL. The ontology code resulting from this activity includes, in addition to the ontology classes, properties, and axioms. Furthermore, following the FAIR principles [10], the ontology code also includes ontology metadata, such as the creator, the title, the publisher, the license and the version of the ontology, in addition to the metadata for each ontology.

To manage the ontology versions developed in the COGITO project, the following version convention was adopted. Following this convention, each release will follow the pattern major, minor, fix, where each field follows the rules:

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

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

2.2.3 Ontology Evaluation

Once the ontology is encoded, it should be evaluated before its online publication. The development of the ontology must guarantee the following aspects:

- The ontology is consistent;
- The ontology does not have syntactic, modelling, or semantic errors;
- The ontology fulfils the requirements scheduled for the ontology, to ensure that the ontology is completed considering the domain experts needs.

2.3 Ontology Publication

During the ontology publication activity, the ontology development team provides an online ontology that is accessible both as a human-readable document and as a machine-readable file from its URI.

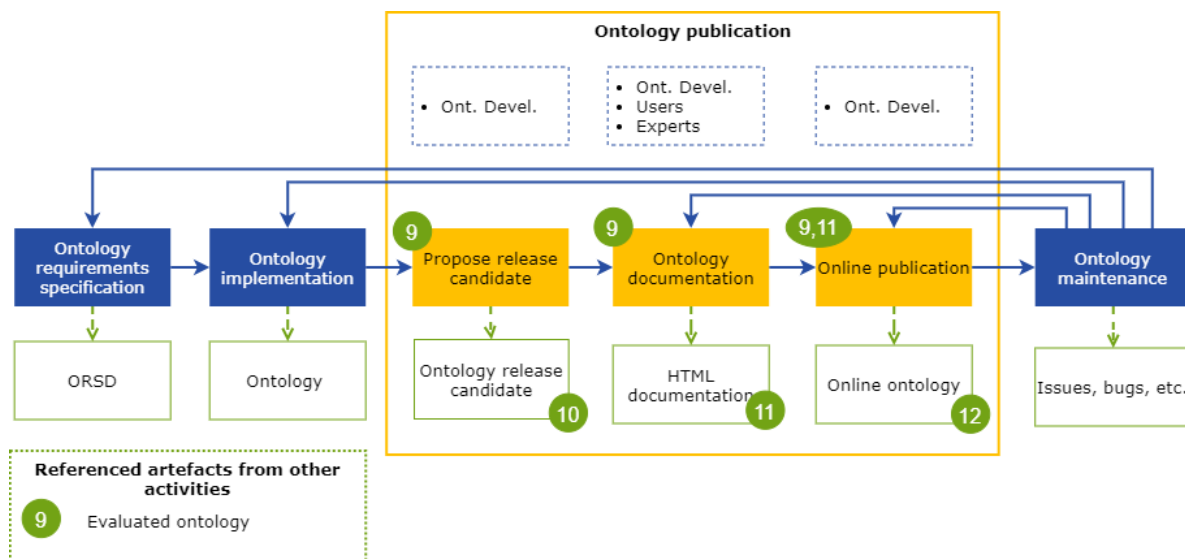


Figure 4: Workflow for the ontology publication activity

Figure 4 shows an overview of the steps performed during the ontology publication activity.

2.3.1 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 documenting such version of the ontology for any other reason is needed (e.g., a set of requirements are fulfilled triggering a new release). In this case, the version at hand becomes a release candidate. In case the ontology is not selected as a 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.

2.3.2 Ontology Documentation

Taking the ontology generated in the previous activities as input, the ontology development team generates the ontology documentation. This documentation includes the following:

- An HTML description of the ontology that describes the classes, object properties, and data properties of the ontology, the license URI, and title being used. Domain experts must collaborate with the ontology development team to describe classes and properties. This description also includes metadata such as creator, publisher, date of creation, last modification, or version number. It also includes links to different formats for serialisation of the ontology, such as TTL, JSON-LD, or RDF/XML.
- Diagrams that store the graphical representation of the ontology, including taxonomy and class diagrams.

2.3.3 Online Publication

During this activity, the ontology, which should already be validated and documented, is published on the Web. This ontology is accessible through its URI as a machine-readable and human-readable file using content negotiation.

2.4 Ontology Maintenance

During this activity, the ontology is updated, and new requirements can be proposed for inclusion in the ontology. Furthermore, during this activity, the ontology development team, together with domain experts and users, can identify and correct errors in the ontology.

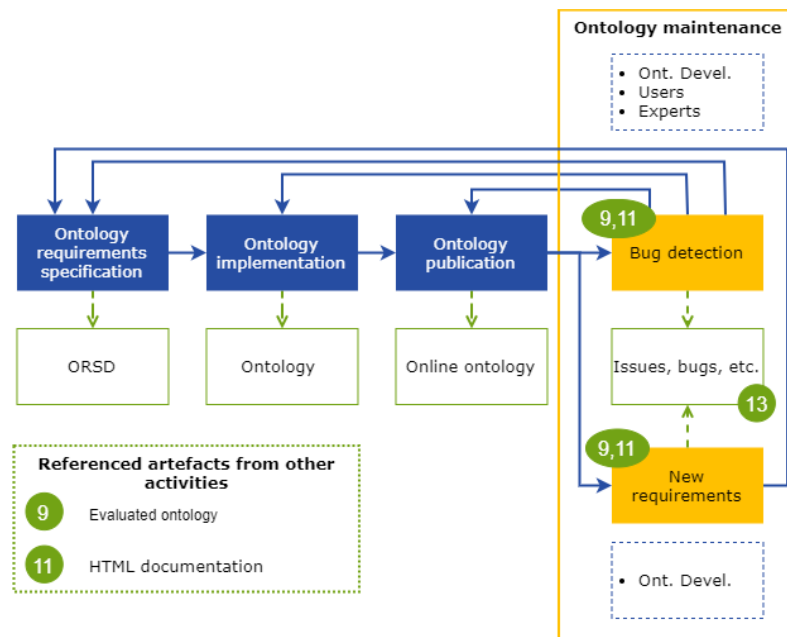


Figure 5: Workflow for the ontology maintenance activity

Figure 5 shows the steps to follow in the maintenance activity of the ontology.

2.4.1 Bug Detection

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

2.4.2 New Requirements Proposal

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

3 Ontology Requirements Specification

Since the goal of the ontology is to facilitate the sharing and interoperability of data between the several COGITO components through the Digital Twin Platform (see Table 1), we decided to analyse each component of the architecture, defined in D2.5 [5], in order to gather information about the data it needs as input from other components and the data it produces as output to other components.

Several meetings with the partners involved in the development/usage of the components have helped us specify the requirements that are aligned with the needs of those components. These requirements have been organised into three different domains: the facility, the construction process, the resources used in that process and the quality of the resulting facility. Table 2 presents the list of requirements for the domains we have identified so far.

Table 1: General information on the specification of ontology requirements

Purpose	To facilitate data sharing and interoperability among the COGITO components through the Digital Twin Platform
Scope	Limited to the data shared among the COGITO components through the Digital Twin Platform
Implementation language	Web Ontology Language (OWL)
Intended end-users	COGITO components and application developers COGITO end users and stakeholders
Non-functional ontology requirements	Annotated in English Linked to standards when possible Open license Available online
Ontology functional requirements	Detailed information in the COGITO ontology portal

Table 2: List of requirements per domain

Identifier (domain+id)	Competency Question / Fact
FACI-1	A project is related to a site and one or more processes
FACI-2	A project can have images and point clouds
FACI-3	A site contains a facility
FACI-4	Facilities include buildings, railways, bridges and roads
FACI-5	Buildings can have storeys, while railways, bridges and roads can have facility parts
FACI-6	Buildings, storeys, spaces and facility parts can contain elements and spatial zones
FACI-7	Spatial zones can contain elements
FACI-8	Elements include walls, slabs, columns, etc.
FACI-9	An element can have sub-elements
FACI-10	Spatial zones include tracked zones (used by tags) and construction zones (which can be walkable or fall zones and can have safety information)
PROC-1	A (project) process has a cost, which is measured in a certain currency
PROC-2	A task belongs to a certain process
PROC-3	A task is related to one or several (construction) elements
PROC-4	A task includes progress and priority information

PROC-5	A task can have a date of creation, beginning and end
PROC-6	A task can have sub-tasks
PROC-7	A task can have predecessor tasks
PROC-8	A task requires certain resource types
PROC-9	A work order is related to a process
PROC-10	A work order has resources assigned
RESO-1	Human worker, equipment, and tracking tags are resources
RESO-2	A resource is a spatial thing
RESO-3	A resource belongs to a resource type
RESO-4	Types of resources include roles, equipment type, and tracking tag groups
RESO-5	Tag groups include human and equipment
RESO-6	Humans and equipment can have a tracking tag assigned
RESO-7	A resource type has an identifier, a name, the maximum quantity and a cost per hour
RESO-8	A resource has an identifier and a status
QUAL-1	An element can have a defect
QUAL-2	A defect has a type of defect and a severity indicator
QUAL-3	A defect can be detected, confirmed or not confirmed
QUAL-4	An image is taken at a time and location
QUAL-5	When an image has been processed, its metadata includes the type of processing
QUAL-6	An element can have geometric quality information
QUAL-7	A piece of geometric quality information is related to a rule and includes a value (passed or not passed) and a specification of the potential issue(s)
QUAL-8	Tags in a project have an estimated accuracy
QUAL-9	A zone and an element can be related to a safety information

4 Overview of the COGITO Ontology Network

The ontology network developed for the COGITO project currently consists of four ontology modules, each module corresponding to one specific domain, namely, facility⁴, process, resource and quality. Figure 6 provides a graphical overview of the COGITO ontology network showing the main concepts defined in each module. To understand this figure and the ones provided in the next subsections, each providing details for a particular module, it is important to take into account the following paragraphs.

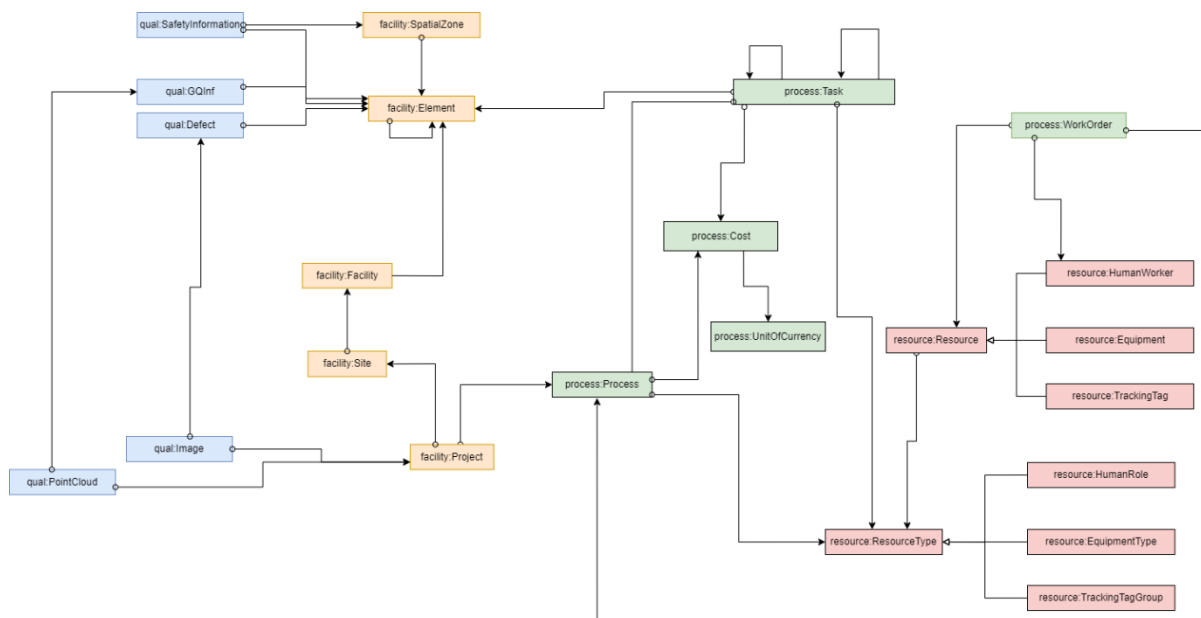


Figure 6: Overview of the COGITO ontology network

Coloured boxes are used to represent modules currently implemented, while white boxes represent reused terms from existing ontologies. As can be seen, some of the ontologies reviewed in previous steps of the project were reused according to the results of the process described in this section, like BOT. Additional cross-domain ontologies, such as WGS84, were also reused once requirements and conceptualisations were analysed in detail. The prefixes, and corresponding ontologies, created and reused in the COGITO ontology are listed in Table 3 and Table 4, respectively.

Table 3: List of the ontologies created in the COGITO project

Prefix	Namespace
facility	https://cogito.iot.linkeddata.es/def/facility#
process	https://cogito.iot.linkeddata.es/def/process#
qual	https://cogito.iot.linkeddata.es/def/quality#
resource	https://cogito.iot.linkeddata.es/def/resource#

Table 4: List of the ontologies reused in the COGITO project

Prefix	Namespace
beo	https://pi.pauwel.be/voc/buildingelement#
bot	https://w3id.org/bot#
geo	http://www.w3.org/2003/01/geo/wgs84_pos
td	https://www.w3.org/2019/wot/td#

⁴ Note that this module was called “construction” in D3.2

The main hierarchies between concepts are also included. These hierarchies are represented by arrows with white endings (triangles) and can be read as follows: the class in the origin of the arrow is a subclass of the class at the end of the arrow. Ad hoc relations between different modules and within modules are also present. Arrows are used to represent these properties between classes and to represent some RDF, RDFS, and OWL constructs. More precisely:

- **Plain arrows with white triangles** represent the subclass relationship between two classes. The origin of the arrows is the class to be declared as a subclass of the class at the destination of the arrow.
- **Plain arrows between two classes** indicate that the object property has declared as domain the class in the origin, and as range the class in the destination of the arrow. The identifier of the object property is indicated in the arrow.
- **Dashed labelled arrows** between two classes indicate that the object property can be instantiated between the classes in the origin and the destination of the arrow. The identifier of the object property is indicated in the arrow.
- **Dashed arrows with the identifiers between stereotype signs** (i.e., “<< >>”) refer to OWL constructs that are applied to some ontology elements, that is, they can be applied to classes or properties depending on the OWL construct being used.
- **Dashed arrows without identifiers** are used to represent the `rdf:type` relation, indicating that the element in the origin is an instance of the class in the destination of the arrow.

Datatype properties are denoted by rectangles attached to the classes, in a UML-oriented way. Dashed boxes represent datatype properties that can be applied to the class it is attached to, while plain boxes represent that the domain of the datatype property is declared to be the class attached.

For more details about the graphical notation used in this diagram, you can see the Chowlk Visual Notation⁵.

The complete and up-to-date documentation of each ontology module is provided online, as explained in Section 2. In the rest of this section, only the main concepts and modelling decisions are detailed.

4.1 Facility Module

The current conceptual model defined for the Facility module is shown Figure 7.

The classes and properties used to describe the topological concepts of a building in this module are based on the Building Topology Ontology (BOT), while other construction products, such as railways, bridges, and roads are described by new classes and properties.

- `facility:Site` is defined as a subclass of `bot:Site` and as such is a part of the physical world or a virtual world that is inherently both located in this world and having a 3D spatial extent. It contains (`bot:containsZone`) one or more `facility:Facility`.
- `facility:SpatialZone` is defined as a subclass of `bot:Zone` and as such is a part of the physical or a virtual world that is inherently both located in this world and has a 3D spatial extent. This class is the root of a hierarchy that includes `facility:ConstructionZone` used to represent zones used by the Health and Safety module and `facilityTrackedZone`, used to represent zones used by the IoT preprocessing module.
- `facility:Space` is defined as a subclass of `bot:Space` and as such is a part of the physical world or a virtual world whose 3D spatial extent is bounded physically or theoretically and provides for certain functions within the zone it is contained in.
- `facility:Facility` is defined as something designed and built to serve a specific function providing a convenience or a service. This new class includes `facility:Railway`, `facility:Bridge`, `facility:Road` and `facility:Building`. It is a class that is not considered in BOT, but we can include it in the `bot:Zone` hierarchy.
- `facility:FacilityPart` is defined as something that is contained (`facility:hasFacilityPart`) in a `facility:Facility`. A `facility:FacilityPart` can contain subfacility parts (`facility:hasSubFacilityPart`).

⁵ https://chowlk.linkeddata.es/chowlk_spec

- `facility:Storey` is defined as a subclass of `bot:Storey` and as such, it is contained (`bot:hasStorey`) in one `facility:Building` and is intended to contain (`bot:hasSpace`) one or more `facility:Space` that are horizontally connected.
- `facility:Element` is defined as a subclass of `bot:Element` and as such, a constituent of a construction entity with a characteristic technical function, form, or position. A `facility:Element` can contain sub-elements (`bot:hasSubElement`). A `facility:SpatialZone` can contain (`bot:containsElement`) some `facility:Elements` (and so do `facility:FacilityPart`, `facility:Space` and `facility:Storey`). A `beo:BuildingElement` is a subclass of `facility:Element`. This class involves a `process:Task` and there is information about its visual quality (`qual:Defect`) and geometric quality (`qual:GQInf`).

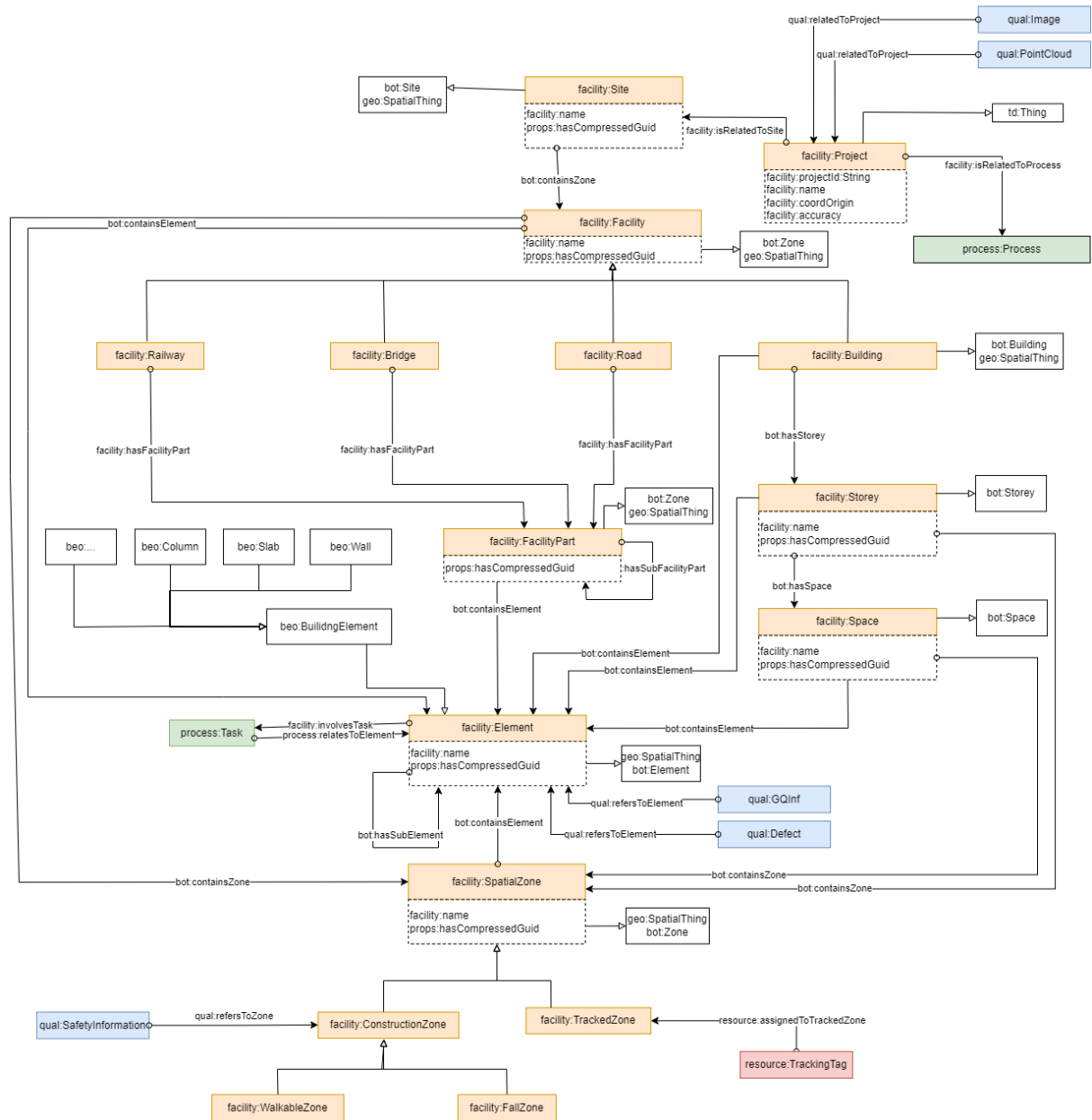


Figure 7: General overview of the COGITO facility ontology

These BOT subclasses have been created because they have specific properties such as `facility:name` and `props:hasCompressedGuid`. They are also subclasses of `geo:SpatialThing` in order to reuse its location properties.

There is a class shared among the four modules: `facility:Project`, which is defined as a large or major undertaking, especially one involving considerable money, personnel, and equipment. It is also a subclass

of `td:Thing` and as such, an abstraction of a physical or a virtual entity -in this case, a project-whose metadata and interfaces are described by a `WoT Thing Description`. A `facility:Project` is related to a `facility:Site` and one or more `process:Process`, and each `qual:Image` and `qual:PointCloud`.

4.2 Process Module

The current conceptual model defined for the Process module is depicted in Figure 8. The main classes and properties in this module have been defined anew:

- `process:Process` is defined as a series of actions aimed at accomplishing some result (in this case, related to a `facility:Project`). It is a synonym for workflow.
- `process:Task` is defined as a piece of work, which is performed in a `process:Process` (`process:belongsTo`) and is related to a `facility:Element`. A `process:Task` can have information about its duration (`process:hasBeginningDate` and `hasEndDate`). We can include the `process:status` and the `process:progress` of a `process:Task`. A task can be assigned (`resource:hasResourceType`) several `ResourceType`,
- `process:Cost` is defined as the price paid to acquire, produce, accomplish, or maintain anything (in this case, `process:Process` and `process:Task`); this price is measured (`process:measuredIn`) in a currency (`process:UnitOfCurrency`).
- `process:WorkOrder` is defined as a command or instruction authorizing specific work, repairs, etc., to be done. It has several `resource:Resource` assigned (`resource:hasAssignedResource`), one of which is a main provider (`resource:hasMainProvider`) and belongs to (`process:belongsToProcess`) a `process:Process`.

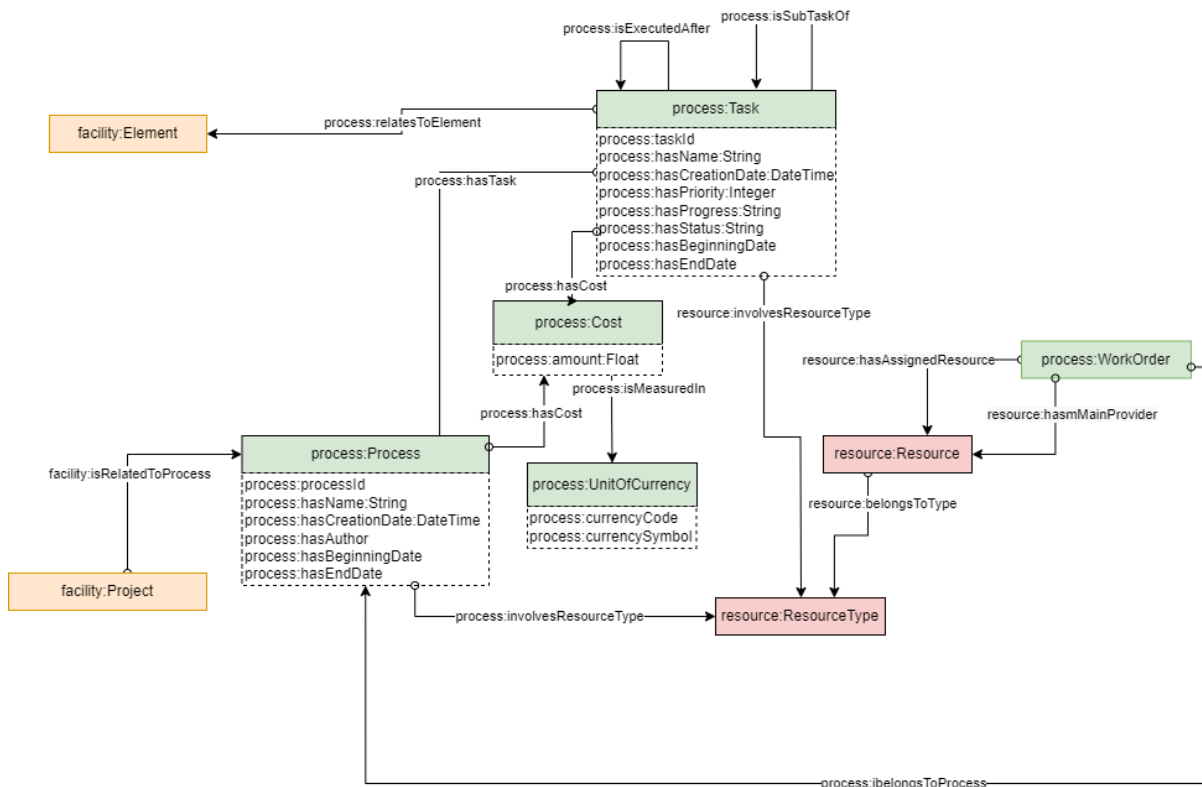


Figure 8: General overview of the COGITO process ontology

4.3 Resource Module

The current conceptual model defined for the Resource module is shown in Figure 9. The main classes and properties of this module are the following:

- `resource:Resource` is defined as a source of supply, support, or aid, especially one that can be readily drawn upon when needed. `resource:HumanWorker`, `resource:Equipment` and `resource:trackingTrack` are subclasses of `resource:Resource`. It is also a subclass of `geo:SpatialThing` in order to reuse its location properties. A `resource:Resource` belongs to a `resource:ResourceType`;
- A `resource:ResourceType` is defined as the kind of resources assigned to a `process:Task` or involved in a `process:Process`, indicating their maximum quantity (`resource: maxUnit`) and cost (`resource:costPerHour`);
- `resource:HumanWorker` is defined as a labourer or employee who plays a role (`resource:HumanRole`) for a `process:WorkOrder`.

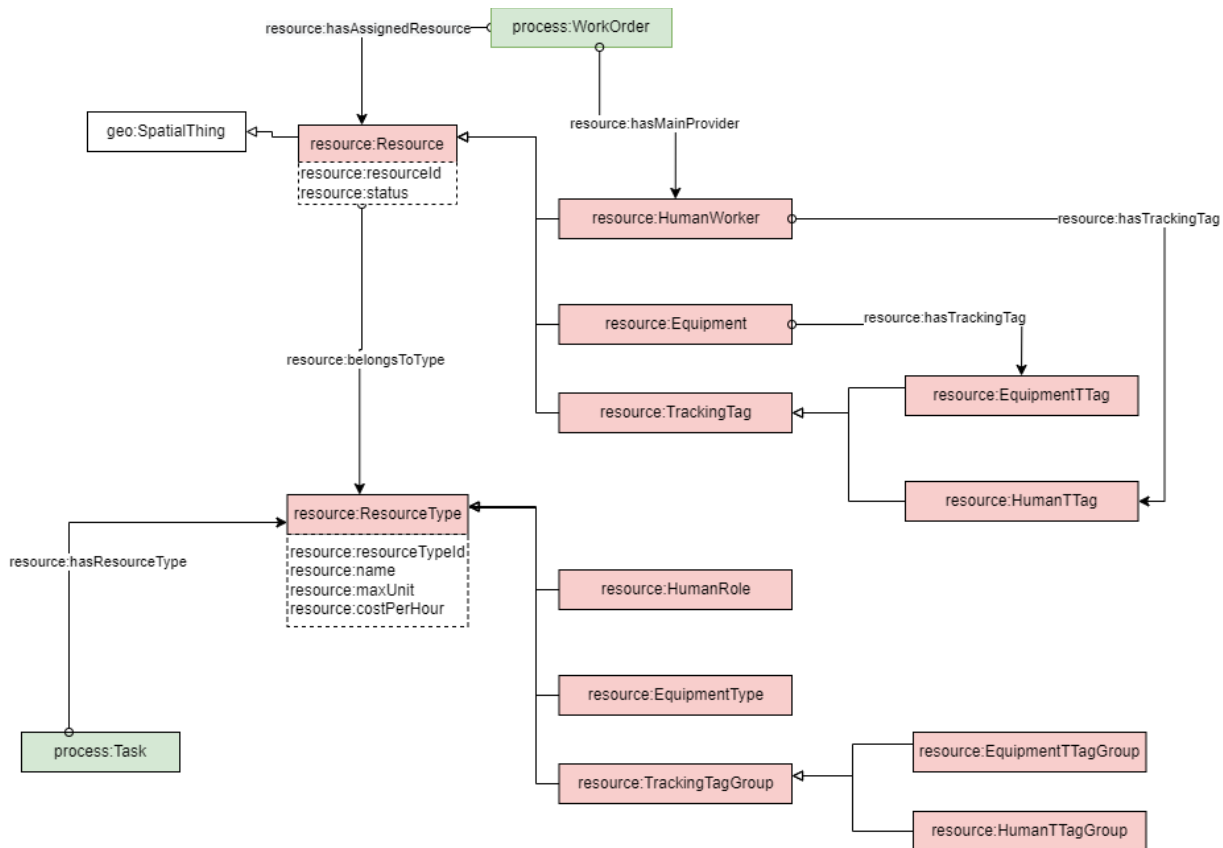


Figure 9: General overview of the COGITO resource ontology

4.4 Quality Module

The current conceptual model defined for the Quality module is depicted in Figure 10. The safety part is currently under development and might lead to a new module in the future. This module's main classes and properties are the following:

- `qual:Defect` is defined as a shortcoming, fault, or imperfection regarding a particular `facility:Element`. This `qual:Defect` is reflected as a `qual:Image`, which can be processed and is taken at a time and location on a kind of material.
- `qual:GeometricQualityInformation` is defined as data informing a particular problem regarding a particular `qual:Rule` on a `facility:Element`. This information is part of a `qual:ListOfGeometricQualityInformation` that comes from analysing a `qual:PointCloud`.
- `qual:SafetyInformation` is defined as data to prevent injuries by taking into account the characteristics of a `facilityConstructionZone`.

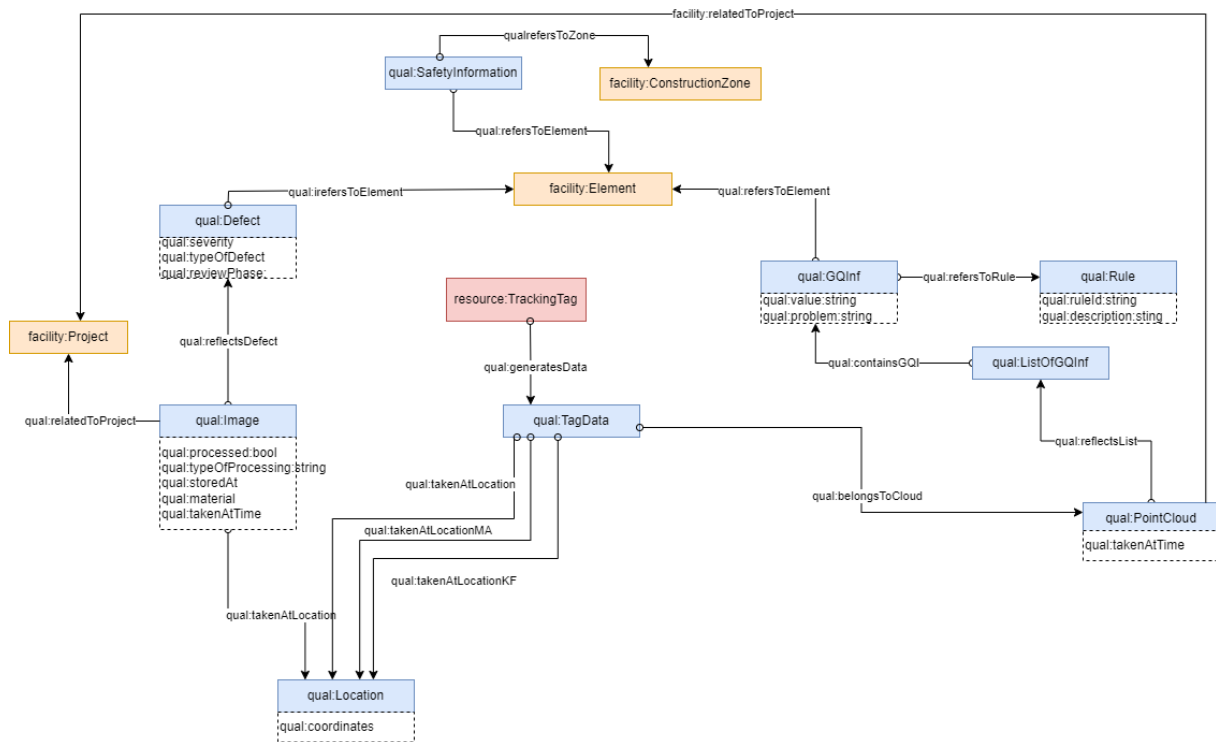


Figure 10: General overview of the COGITO quality ontology

5 Ontology Development Infrastructure

This section describes the ontology development infrastructure used to support the activities in the ontology development process described above.

5.1 Infrastructure to Support the Requirements Specification Activity

The ontology development team used spreadsheets to store requirements per component. An excerpt of these requirements is shown in Figure 11.

Identifier (component+id)	Competency Question / Natural language sentence (fact)	Answer	Status (Proposed, Accepted, Rejected, Pending, Deprecated)	Superseded by	Comments	Extracted from (provenance)	Priority (High, Medium, Low)
pms-1	A construction can contain one or several zones					minutes of meeting on PMS	
pms-2	A construction can have one or several spaces					minutes of meeting on PMS	
pms-3	A zone can have one or several spaces					minutes of meeting on PMS	
pms-4	Resources can be allocated to a task					minutes of meeting on PMS	
pms-5	Workers, equipment and materials are resources					minutes of meeting on PMS	
pms-6	A resource is a spatial thing					minutes of meeting on PMS	
pms-7	A (construction) element has quality information					minutes of meeting on PMS	
pms-8	A (construction) element is a spatial thing					minutes of meeting on PMS	
pms-9	A task has quality information					minutes of meeting on PMS	
pms-10	A task is related to one or several (construction) elements					minutes of meeting on PMS	

Figure 11: COGITO ontology requirements for PMS

These requirements per component were reorganised per domain (see Section 3) and converted into an HTML file and uploaded to the COGITO ontology portal⁶ with the most relevant information for users to facilitate visualisation. In Figure 12 a snapshot of the HTML documentation of the COGITO requirements is presented.


Ontology requirements							
							
Here you can find the list of the requirements identified for the COGITO Construction Process ontology and their main features.							
Identifier [†]	Component	Competency Question / Natural language sentence (fact) [‡]	Answer	Status [†]	Superseded by	Priority [‡]	
PROC-1	WODM	A task has quality information					
PROC-2	WODM	A task is related to one or several (construction) elements					
PROC-3	WODM	A (project) process has a cost, which is measured in a certain currency					
PROC-4	WODM	A task belongs to a certain process					
PROC-5	WODM	A task includes progress information					
PROC-6	WODM	A task can have a date of creation					
PROC-7	WODM	A task can have an order in the workflow					
PROC-8	WODM	A task has requirements regarding resources					
PROC-9	PMS	Resources can be allocated to a task					

Figure 12: HTML requirements for the construction process

⁶ <https://cogito.iot.linkeddata.es/>

5.2 Infrastructure to Support the Ontology Implementation Activity

To support the implementation activity, the ontology development team uses several tools to edit, store, and evaluate the ontology.

- As ontology editor the ontology development team uses Protégé⁷ which allows the creation, visualisation, and manipulation of ontologies.
- For ontology storage, the ontology development team uses GitHub⁸. A GitHub repository is created for each ontology in the COGITO ontology network. Each repository includes:
 - A folder with the implementation of the ontology.
 - A folder with the ontology modelling diagrams.
 - A folder with the documentation of the ontology.
 - A folder with the requirements and tests of the ontology.

The development team uses the OnToology⁹ tool to generate documentation and evaluate the ontology. OnToology, which integrates the tools Widoco¹⁰ [11] and OOPS!¹¹ [12], automatically generates a folder in the GitHub repository that includes the resources: diagrams, documentation, and evaluation report.

5.3 Infrastructure to Support the Ontology Publication Activity

To support the publication activity, the ontology development team created an online ontology portal to make the ontology and all the associated information (repository, requirements, tests, releases, etc.) available to the users. This ontology portal has different sections:

- Ontologies
- How we work

5.3.1 Ontologies

The Ontologies section, which is the primary section of the portal, shows the main information about the ontologies created in the COGITO ontology network. Figure 13 shows an overview of the information exposed in this section of the portal. The section follows a tabular approach that includes the following:

- Link to the ontology documentation published on the Web
- Ontology Description
- Link to each GitHub repository
- Link to each GitHub issue tracker
- HTML description of the requirements identified by the domain experts
- Link to each ontology release

⁷ <https://protege.stanford.e>

⁸ <https://github.com/orgs/oeg-upm/teams/cogito>

⁹ <https://ontology.linkeddata.es/>

¹⁰ <https://github.com/dgarijo/Widoco>

¹¹ <http://oops.linkeddata.es/>

Ontologies

Ontology testing



Here you can find the list of ontologies developed for COGITO project

If you want to contribute developing ontologies please follow the [guidelines](#) we provide

Ontology	Description	Requirements	Repository	Issue tracker	Releases
COGITO Construction Process ontology	This ontology aims to model the construction process in the COGITO ontology	Ontology Requirements	Ontology Repository	Ontology Issue Tracker	Ontology Releases
COGITO Construction ontology	This ontology aims to model the construction data exchanges in the COGITO project	Ontology Requirements	Ontology Repository	Ontology Issue Tracker	Ontology Releases
COGITO Resources ontology	This ontology aims to model the resources in the COGITO project	Ontology Requirements	Ontology Repository	Ontology Issue Tracker	Ontology Releases

Figure 13: Ontology section in the COGITO ontology portal

5.3.2 How We Work

The section “How We Work” provides a brief overview of the proposed process for developing ontologies as well as certain guidelines which should be useful to anyone who wants to contribute to the ontologies. This section includes information about the following.

- How should the repository be structured
- Tools recommended to be used during the ontology development process
- Ontology versioning
- Management of issues

5.4 Infrastructure to Support the Ontology Maintenance Activity

To provide support for maintenance activities, ontology developers use the GitHub issue tracker, which manages and maintains the list of issues identified by domain experts and ontology developers. The GitHub issue tracker provides the status of the issue, assignee, and description and allows one to add comments to the issue to discuss about it. Each issue tracker is associated with a GitHub repository and consequently with an ontology in the COGITO ontology network.

To manage changes in the ontology, all new proposals and improvements must be agreed upon by all members of the ontology development team. If domain experts, users or ontology developers want to add, delete or modify concepts in the ontology, they must create a new issue in the GitHub issue tracker associated with the ontology to be modified which will then be used to discuss the approval or rejection of the proposal. Figure 14 shows a snapshot of the GitHub issue tracker for one of the COGITO ontologies.

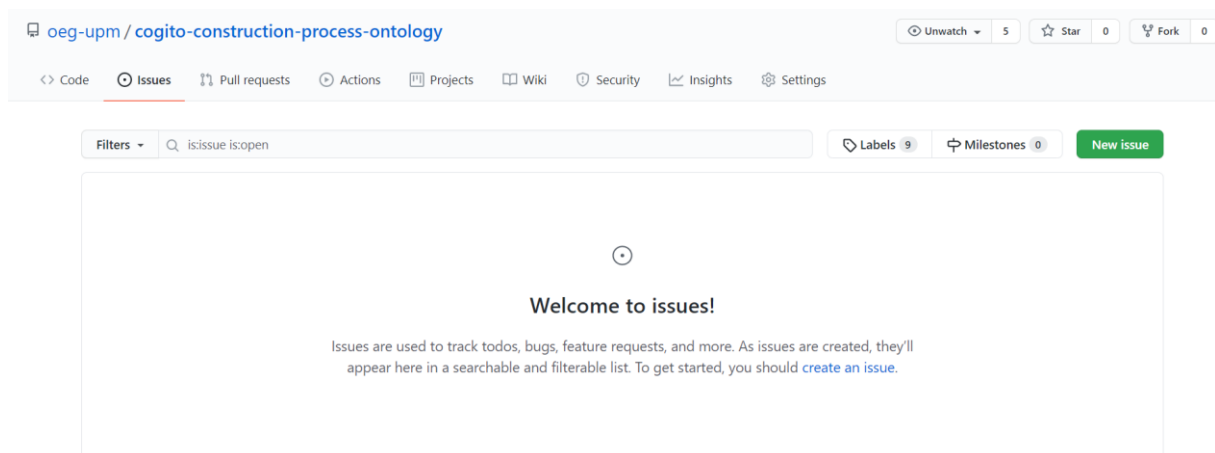


Figure 14: Snapshot of the GitHub issue tracker

6 Conclusions

This document provides details on the methodology and technological infrastructure used to develop the COGITO ontology network, as well as the on the second version of the ontologies that make up such an ontology network. The development of these ontologies is aligned with the collection of COGITO requirements as well as with the specification of the COGITO architecture; since this is an ongoing task, the ontology network will evolve over time.

The current version of the COGITO ontology network (see Figure 6) consists of four modules, corresponding to the facility (the construction itself), the construction process, the construction resources and the quality achieved (see Table 3). These modules have already been conceptualised, implemented and published in the second version of the COGITO ontologies and will be modified and complemented with new modules as the project progresses.

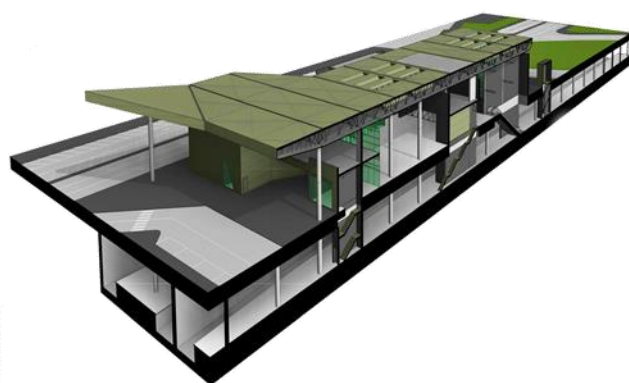
In D3.2 [2] a coverage analysis was carried out identifying to what extent terms appearing in the requirements also appeared in the ontologies described in the previous D3.1 deliverable [3]. The conclusions highlighted the fact that reusing ontologies requires analysing multiple factors beyond coverage and some of them are not easily quantifiable or cannot even be measured objectively. For this reason, another coverage analysis for the new ontology requirements was not executed.

The COGITO ontology network will allow for the sharing and interoperability among the different components of the COGITO architecture, especially in the communications with the Digital Twin platform. In this context, this document and the future versions of the ontology, will be particularly important when developing each component of the COGITO architecture, i.e., the future results of work packages 4, 5, 6, and 7 and their integration in work package 8.

The next steps will be devoted to continuing the collection of more ontological requirements from the pilots and the architecture in order to iteratively evolve the COGITO ontology network. The COGITO ontology portal will serve as a living repository for the different ontologies and will contain the latest version of the developed ontologies and all related artifacts.

References

- [1] M. Poveda-Villalón, A. Fernández-Izquierdo and R. García-Castro, "Linked Open Terms (LOT) Methodology," January 2019. [Online]. Available: <http://doi.org/10.5281/zenodo.2539305>.
- [2] "D3.2. COGITO Data Model and Ontology Definition and Interoperability Design," 2021.
- [3] "D3.1- Survey of Existing Models, Ontologies and Associated Standardization Efforts," 2021.
- [4] "D2.1 - Stakeholder Requirements for the COGITO System," 2021.
- [5] "D2.4- COGITO System Architecture," 2021.
- [6] M. C. Suárez-Figueroa, A. Gómez-Pérez and M. Fernández-López, "The NeOn Methodology for Ontology Engineering," in *Ontology engineering in a networked world*, Springer, Berlin, Heidelberg, 2011.
- [7] R. García-Castro, A. Fernández-Izquierdo, C. Heinz, P. Kostelnik, Poveda-Villalón-María and F. Serena, "D2.2. Detailed Specification of the Semantic Model," 2017.
- [8] S. Chávez, F. Bosché, N. Bountouni, S. Fenz, R. García-Castro, G. Giannakis, S. González-Gerpe, S. Kollmer, S. Kousouris, D. Ntalaperas, T. Thanos, F. Lampathaki, M. Poveda-Villalón, E. Valero, D. Vergeti and F. Tavakolizadeh, "D4.2 BIMERR Ontology & Data Model," 2020.
- [9] A. Fernández-Izquierdo, A. Cimmino, R. García-Castro, M. Poveda-Villalón, S. Terzi and C. Patsonakis, "T1.3 DELTA Ontology and Data Modelling Framework," 2019.
- [10] D. Garijo and M. Poveda-Villalón, "Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web," in *Applications and Practices in Ontology Design, Extraction, and Reasoning*, IOS Press, 2020.
- [11] D. Garijo, "WIDOCO: a wizard for documenting ontologies," in *International Semantic Web Conference*, 2017.
- [12] M. a. S.-F. M. C. a. G.-P. A. Poveda-Villalón, "Validating ontologies with oops!," 2012.
- [13] ISO 16739, "Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries (ISO 16739:2013)," CEN, 2016.
- [14] ISO 13790, "Energy performance of buildings — Calculation of energy use for space heating and cooling," 2008.
- [15] M. C. Suárez-Figueroa, A. Gómez-Pérez and B. Villazón-Terrazas, "How to write and use the ontology requirements specification document," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2009.
- [16] A. Fernández-Izquierdo and R. García-Castro, "Conformance testing of ontologies through ontology requirements," *Engineering Applications of Artificial Intelligence*, vol. 97, 2021.
- [17] M. H. Rasmussen, P. Pauwels, C. Hviid and J. Karlshøj, "Proposing a Central AEC Ontology That Allows for Domain Specific Extensions," in *Joint Conference on Computing in Construction*, 2017.
- [18] A. Annane, N. Aussenac-Gilles and M. Kamel, "BBO: BPMN 2.0 Based Ontology for Business Process Representation," in *20th European Conference on Knowledge Management (ECKM'19)*, Lisbon, 2019.



COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310