Co-financed by the European Union
Connecting Europe Facility

Water Health Open knoWledge (WHOW)

Linked Open Data Process Design is finalised

Project number	Agreement number: INEA/CEF/ICT/A2019/2063229 Action No: 2019-EU-IA-0089
Project acronym	WHOW
Project title	Water Health Open knoWledge
Project duration	36 months (01/09/2020 – 31/08/2023)
Programme	Connecting Europe Facility (CEF) Telecom
Activity title	Knowledge Graph Definition - Task 3.3 Linked Open Data production process design
Deliverable number	3.2 [Milestone #5]
Version (date)	1.0 (2022-06-14)
Due date	2022-05-31
Responsible organisation	National Council of Research (ISTC-CNR)
Editors	ISTC-CNR, Aria SpA, ISPRA

Abstract	**This deliverable introduces the design of the overall linked open data process to be implemented at each data provider side.**
Keywords	**Technical architecture, linked open data production process, data transformation, data quality verification**

Editor(s)

Anna Sofia Lippolis, Giorgia Lodi, Andrea Giovanni Nuzzolese (Institute of Cognitive Sciences and Technologies of the Italian National Research Council - ISTC-CNR).

Elena Madiai, Gianluca Carletti (ARIA SpA)

Elio Giulianelli, Giulio Settanta, Marco Picone (ISPRA)

Reviewers

Annalisa Minelli (ISPRA)

Adalberto Sangalli (ARIA SpA)

Acknowledgement

This work was partially supported by the European Commission (EC) through the Connecting Europe Facility (CEF) programme under the WHOW project WHOW (grant agreement no. INEA/CEF/ICT/A2019/2063229).

Change Log

Version	Date	Organisation	Description
0.1	2022-03-31	ISTC-CNR	Creation of the Table of Content
0.2	2022-04-21	Aria SpA, ISPRA	Preliminary content of section 2
0.3	2022-04-28	Aria SpA, ISPRA	Revision of content of section 2
0.4	2022-05-05	ISTC-CNR	Introduction of some design figures for section 3
0.5	2022-05-26	Aria SpA, ISPRA	Revision of section 2 based on reviewer comments
0.6	2022-06-06	ISTC-CNR	Some content for section 3, executive summary, introduction and conclusion
0.7	2022-06-09	ALL	Revisions in section 2 and 3
0.8	2022-06-13	ISTC-CNR	Creation of section 4 based on content of section 3
1.0	2022-06-14	ALL	Final version ready

Table of contents

List of Tables

List of Figures

Executive Summary

In order to build an open and distributed knowledge graph, that is capable of integrating and standardising heterogeneous data of the environmental and health domains, it is necessary to clearly design its production process, which potentially involves different steps as we have already discussed in the WHOW Linked Open Data Reference Architecture [10].

In doing so, some important principles are to be considered. First of all, the Linked Open Data (LOD) process must be sustainable over time and keep on operating even at the end of the WHOW project at each data provider. In this regard, minimising any manual interventions for the knowledge graph production starting from datasets already available, open or not, is a fundamental need. Secondly the process should be designed in order to accommodate different architectural scenarios; that is, the design should be flexible enough so that to be adopted by potentially other data providers, owning and managing various open data infrastructures with datasets supporting the WHOW identified use cases.

Based on these crucial needs, this deliverable represents another pillar of the WHOW project; in fact, it focuses on the design of the whole WHOW LOD production process to be used at data providers and describes:

- the current open data systems that are deployed at ISPRA and Aria SpA. This is necessary to understand how the WHOW process could be successfully integrated in existing infrastructures;
- the design of the WHOW LOD process, presenting both high level views of it and more detailed ones that leverage the UML language notation. In this latter case, activities diagrams are presented to describe all the phases of the process;
- an assessment of the possible tools that can be used to realise the actual transformation of input datasets into a semantic knowledge graph represented through the standard RDF;
- a possible declarative solution, to be integrated in data provider systems, that exploits an OWL meta-level ontology to provide declarative descriptions of a workflow to be executed by a workflow management system. This promising solution, already adopted in other European pilot projects in which ISTC-CNR researchers participated, can provide guarantees on the sustainability of the overall process with minimal manual interventions.

1 Introduction

This is deliverable "3.2 - Linked Open Data Process Design is Finalised". It is the result of the activities conducted in the context of task 3.3 of Activity 3 related to "Knowledge Graph Definition". It extends deliverable 3.1 "Data pre-processing is finalised" with peculiar information about the design of the linked open data generation process.

1.1 Project Overview

The WHOW project aims to foster the creation of the first open and distributed European knowledge graph on water consumption and quality, health parameters and dissemination of diseases to be reused for advanced analysis and development of innovative services.

The project leverages the Linked Open Data paradigm. Water related datasets from Italy and other European countries and Copernicus (the European Union's Earth observation programme) will be used to support the construction of WHOW's knowledge graph, intended as a federation of knowledge graphs deployed at each data provider willing to join the WHOW community. The knowledge graph will be documented on data.europa.eu, the official portal for European data, thanks to the adoption of shared metadata models such as DCAT-AP[1] and its extensions that are relevant for the type of data treated in WHOW (e.g., GeoDCAT-AP[2]). Selected health related datasets mainly from Italy will be linked to specific water datasets.

WHOW targets identified use cases for the creation of its knowledge graph. In order to evaluate such use cases relevant a set of indicators for Sustainable Development Goals (SDGs) are identified along with Key Performance Indicators for metadata and data quality. A co-creation programme, where interested stakeholders and users are engaged from the initial phases of the project, is set-up so as to consider real needs of data re-users.

The initiative supports the Public Open Data Digital Service Infrastructure by helping to boost the development of information products and services based on the re-use and combination of environmental data and health data on disease dissemination.

1.2 Objectives

This document contains the design of the WHOW Linked Open Data process that is to be deployed at each WHOW data provider in order to produce the ultimate WHOW knowledge graph. More specifically, this deliverable details the design of the data preparation, and part of the knowledge graph service management layers presented in the architecture deliverable [10]. The design has been defined so as to provide data providers with a highly extensible and sustainable LOD process that can be potentially adopted by different providers with various open data infrastructures, or with not open data systems at all.

1

https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/11

2 https://semiceu.github.io/GeoDCAT-AP/drafts/latest/

Thanks to the design we propose in this deliverable, WHOW is capable of defining a sustainable pipeline for open knowledge graph production that guarantees authoritativeness, timeliness, semantic accuracy and consistency data quality characteristics, as well as compliance of metadata with the European DCAT-AP profile and related national and thematic extensions.

The target of this document are primarily data providers who want to use the process here proposed for Linked Open Data production purposes.

It is worth highlighting that the process we introduce in this deliverable can be adopted for any type of datasets to be transformed in a semantic knowledge graph, not only for datasets in the water and health sectors.

In summary, this deliverable addresses the specification of the design of the knowledge graph production process and discusses an assessment of possible approaches that can be exploited to achieve such an objective.

1.3 Relationships with other activities

The present deliverable D3.2 is an another important milestone (#5) of the WHOW project. It covers all the activities of task 3.2 of Activity 3 that are linked with other tasks of other activities of the project.

The following Figure 1 shows such relationships.

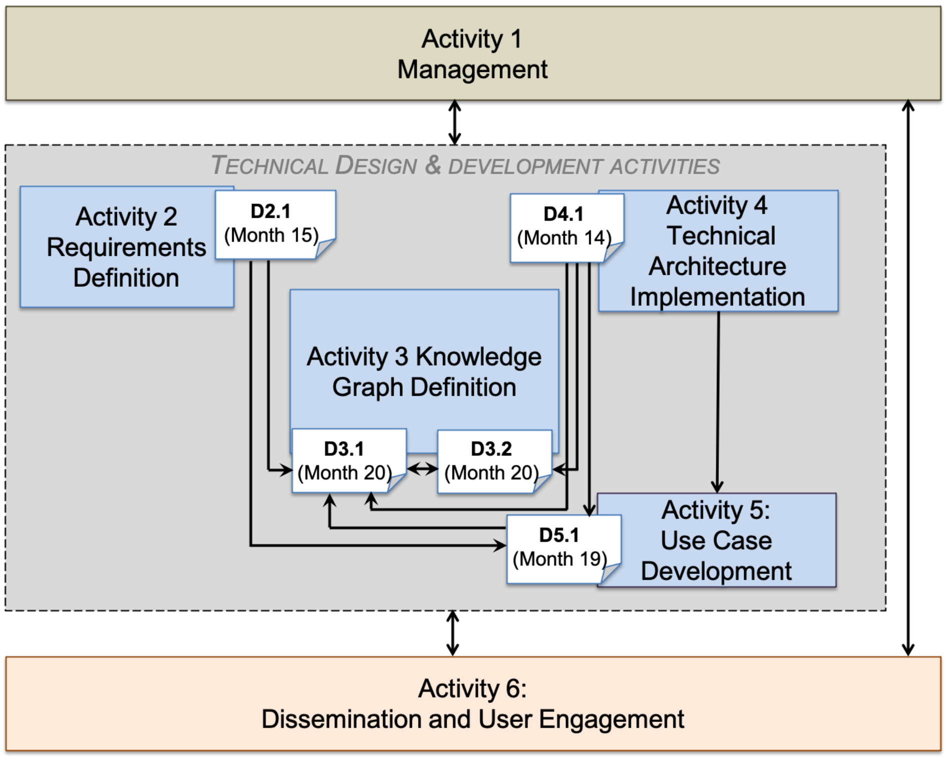


Figure 1: Relationships with other deliverables and activities.

Specifically, the content of this deliverable is strictly dependent on Deliverable 3.1, which covers the data pre-processing. In fact, data pre-processing introduced in D3.1 is a specific phase of the overall process aimed at producing Linked Open Data as the final result. This is also reported in the deliverable on the Linked Open Data Reference Architecture (Deliverable 4.1).

1.4 Structure of the document

This document is organised as follows. In Section 2 existing open data systems are described. More in detail, in Section 2.1 the Aria S.p.A. open data system and in section 2.2 the ISPRA open data system are illustrated. Section 3 describes the Linked Open Data production process and Section 4 the way this process integrates with Aria S.p.A. and ISPRA open data processes. Finally, Section 5 concludes the deliverable.

2 Existing open data systems at data providers

This section reports an analysis of the existing open data systems at ARIA S.p.A. and ISPRA.

2.1 ARIA S.p.A. open data system

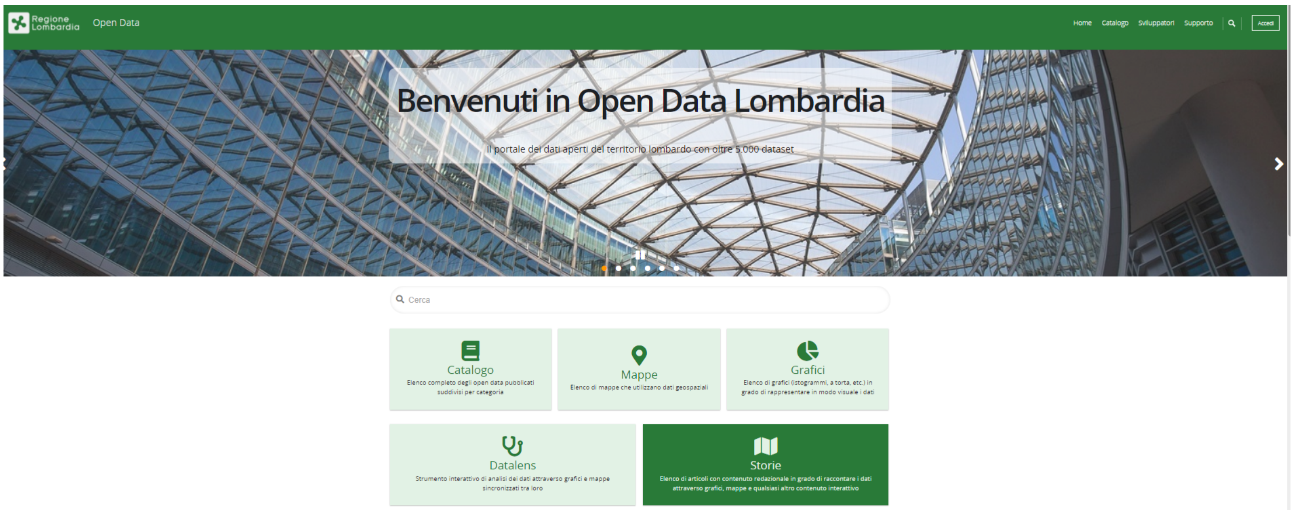
The Regione Lombardia's open data system is currently implemented by ARIA S.p.A.. In the next paragraphs, the reader can find the descriptions of the **Open Data Lombardia Portal** and of the **Publication Process**. Complementary information about data pre-processing for the datasets involved in WHOW use cases implementation is illustrated in the deliverable "Data pre-processing is finalised".

Open Data Lombardia Portal Overview

The Open Data Lombardia Portal, available at the link https://dati.lombardia.it, gathers datasets on a wide range of domains from data providers located in Lombardia. At present, more than 5,000 datasets are published spanning to the following domains: Agriculture, Environment, Manufacturing, Culture, Energy, Family, Education, Transportation, Civil Protection, Health, Public Safety, Sport, Statistics, Open Government and Transparency, Taxes, Tourism, University and Research. Figures about active datasets, unique visitors and downloads are monthly reported and published on the portal, as open data files. In 2021, 4,123,746 visualisations and 1,006,641 downloads were registered. Increasing numbers are expected for 2022.

The main data provider on the portal is Regione Lombardia itself. However, an important contribution in terms of datasets comes from other regional bodies, such as ARPA Lombardia (Regional Agency for Environmental Protection), POLIS Lombardia (Regional Institute for Policy Implementation) and ERSAF (Regional Agency for Agricultural and Forestry Services). On top of that, more than 100 public local agencies and municipalities are publishing datasets on the portal as well.

Born in 2012, the portal is based on the Socrata Connected Government Cloud (SCGC) Software as a Service (SaaS) solution, provided by Tyler Tech Enterprise Data Platform. Figure 2 shows its homepage: users can consult the Catalogue, Maps, Graph generator and other tools which will be described in the next paragraph.



Benvenuti in Open Data Lombardia

Il portale dei dati aperti del territorio lombardo con oltre 5.000 dataset

Elenco completo degli open data pubblicati suddivisi per categoria

Elenco di mappe che utilizzano dati geospaziali

Elenco di grafici (istogrammi, a torta, etc.) in grado di rappresentare in modo visuale i dati

Strumento interattivo di analisi dei dati attraverso grafici e mappe sincronizzati tra loro

Elenco di articoli con contenuto redazionale in grado di raccontare i dati attraverso grafici, mappe e qualsiasi altro contenuto interattivo

Figure 2: Open Data Lombardia Homepage.

Open Data Process

This section describes the open data process. Two actors cooperate to deliver this process: the data owner and the open data team. The data owner is responsible for one or more datasets, and for the pre-processing at the data source level, including the privacy compliance checks and the predefined views production. The open data team is in touch with the data owner, and is responsible for the open data system management, the open data pre-processing and the datasets publication.

By rule, in order to expose a dataset on the open data system for the first time, there are some preliminary activities to be finalised on the Socrata web interface. Firstly, the data owner sends to the open data team a data sample (also a mocked one) exposed in the materialised view as a CSV file. Then through the Socrata backoffice GUI, the open data team uploads the file and defines the data types and adjusts the data formats. Then, Socrata returns a unique ID for the just uploaded dataset as well as API names linked to each variable. Secondly, the data owner shares with the open data team the dataset metadata, through a dedicated metadata sheet, with information on title, description, frequency of the update and last update, data category, licence type. For subsequent changes and integrations, there is an automated process.

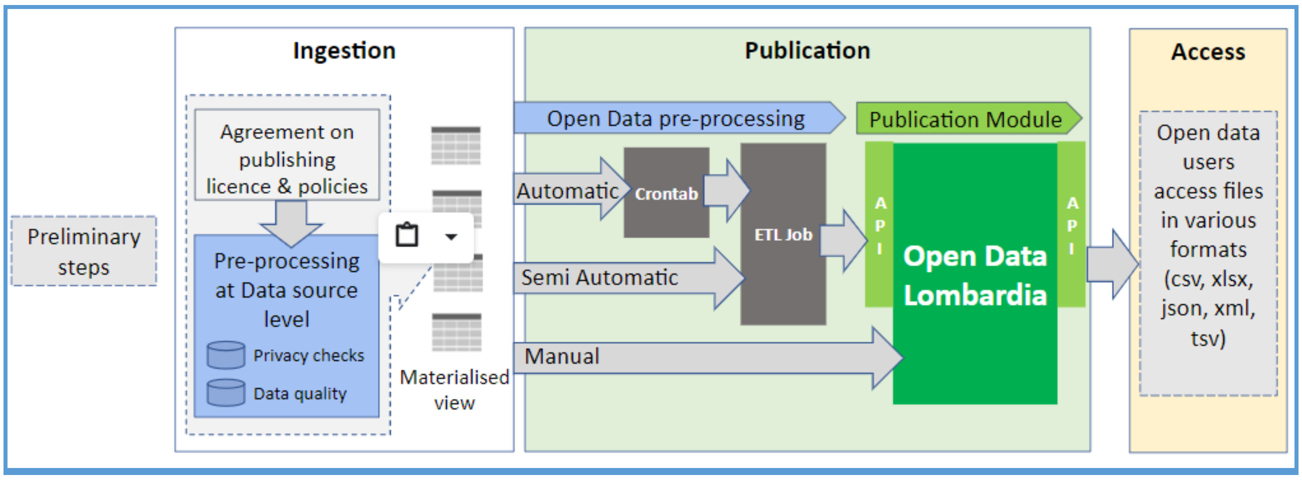


Figure 3: Open Data Process.

As shown in Figure 3, the open data process consists of three macro activities after the preliminary steps: ingestion, publication and access.

Ingestion

Data ingestion is the first macro activity. The data owner and the open data team agree on publishing licence and policies, fixing the conditions about publishing licence, data sources, update frequency, dataset structure and publishable variables.

As a first step, each data owner has the faculty to define its own dataset publishing licence. Table 1 shows the open data licences of the Creative Commons family, widely adopted in the open data context.

Table 1. Creative Commons Scheme for Open Data.

Licence name	Abbreviation	Icon	Attribution required	Allows remix culture	Allows commercial use	Allows Free Cultural Works	Meets the OKF 'Open Definition'
"No Rights Reserved"	CC0	PUBLIC DOMAIN	No	Yes	Yes	Yes	Yes
Attribution	BY	cc BY	Yes	Yes	Yes	Yes	Yes
Attribution-ShareAlike	BY-SA	cc BY SA	Yes and distribution under share alike policy	Yes	Yes	Yes	Yes

CC0 is a public domain or "waiver" where the declarant "openly, fully, permanently, irrevocably and unconditionally waives, abandons all its rights, all related claims, demands, causes of action, whether now known or hereafter unknown (expressly including present claims as well as future claims) relating to the work". Therefore, in the case CC0 is used, all the uses are possible, including commercial ones.

The CC-BY licence allows the dataset to be shared, adapted and created even for commercial purposes with the sole constraint of attributing the authorship of the dataset. Lastly, the CC-BY-SA licence allows the dataset to be shared, adapted and created even for commercial purposes subject to two constraints: a) dataset attribution authorship; b) distribute any derivative works under the same licence as the original work. For the Regione Lombardia datasets, the CC0 licence was chosen[3].

As a second step, the data owner is in charge of pre-processing at the data source level: this is about privacy and data quality checks. It is the data owner's responsibility to check whether the dataset includes personal information, and, in case, to anonymise it. The data owner is responsible as well for data quality and standardisation: geocoding, data cleaning and standardisation are the most common data manipulation tasks as also shown in the Data Preprocessing deliverable.

After that, materialised views are implemented between the data owner and the open data team. They can be views, but also of different sorts (e.g. api, ftp): the goal is to share the information with the open data server, that through an ETL (Extract, Transform, Load) will upload them to the open data portal.

Both publishing rules and predefined views are defined prior to the dataset publication and they can be either maintained or changed throughout the dataset publication period. In this second case, the data

[3] https://creativecommons.org/publicdomain/zero/1.0/deed.it.

owner should inform the open data team when changes are needed, so that the subsequent activity, open data pre-processing, is properly run.

Data Publication

Data publication is the second macro activity. Metadata file and dataset uploading to the portal are the most relevant activities for this step. At this stage, data pre-processing can be performed too, via specific ETL procedures. Metadata file and the dataset update can be performed in three different modes: automatic, semi-automatic and manual.

In *automatic* mode, ETL jobs are run to automate metadata file and dataset preparation and update, by using an open-source software, such as Talend Open Studio[4]. ETL steps are automated via APIs and they are launched using crontab. Figure 4 shows an example of the publication process in automatic mode.

In *semi-automatic* mode, ETL jobs are run to automate metadata file and dataset preparation and update. These activities are launched on demand as crontab is not working in this case. This happens, for instance, when the dataset must undergo a refreshing and updating process: during that period the dataset may be inconsistent and publication must be postponed after the update is completed.

In *manual* mode, metadata and dataset preparation and update are performed manually.

Regardless of which of these three processes is enacted, after its finalisation, the publication module is initialised. The dataset in CSV format and the metadata file in ini format are processed by an ETL that automatically transforms the files in a json file ready to be uploaded to the open data system (the cloud SCGC).

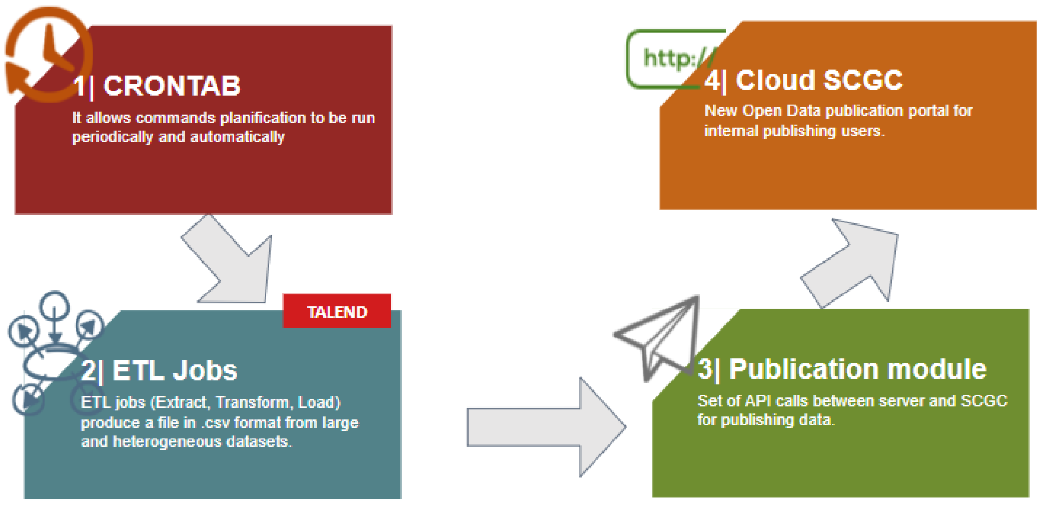

1| CRONTAB
It allows commands planification to be run periodically and automatically
2| ETL Jobs
TALEND
ETL jobs (Extract, Transform, Load) produce a file in .csv format from large and heterogeneous datasets.
3| Publication module
Set of API calls between server and SCGC for publishing data.
4| Cloud SCGC
http://
New Open Data publication portal for internal publishing users.

Figure 4: Publication Process in automatic mode.

[4] Further information is available at this link.

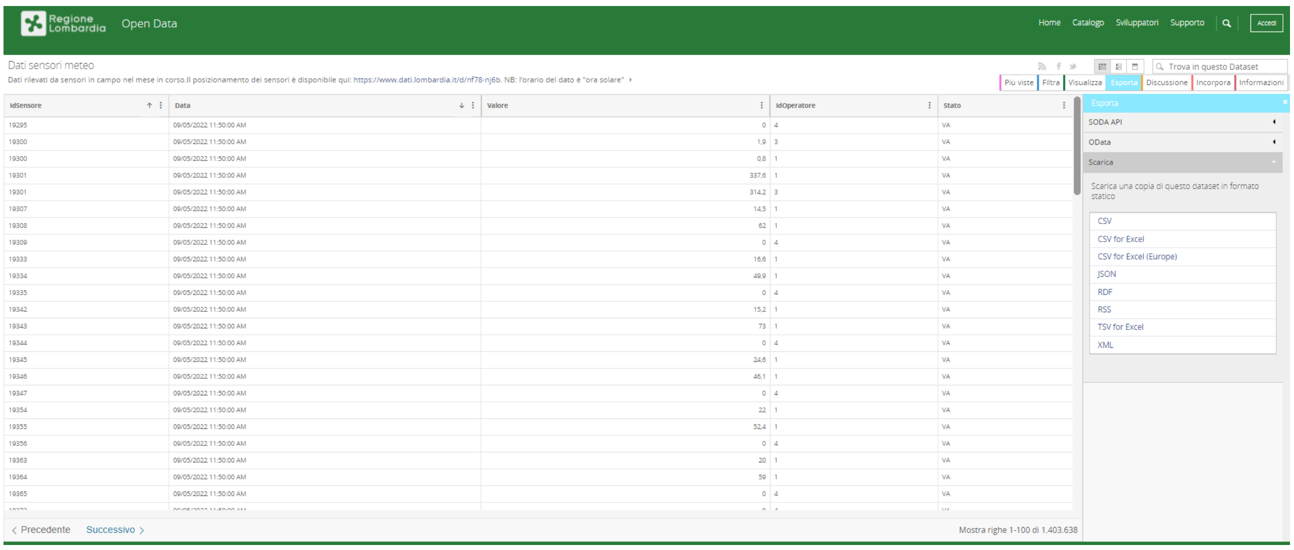
Data access

Data access is the last macro activity of the open data process. Once the data publication is finalised and the dataset is uploaded to the Cloud SGCG, it becomes accessible on the open data portal for consultation, visualisation and download.

Users can search for datasets using the Catalogue, then they can consult the metadata section and download the data in various formats (CSV, json, geojson, rdf…), or access them via APIs. More in detail, the Socrata service provides endpoints to make datasets accessible via APIs. As a next step, it is possible for users to download the dataset in the preferred format among json, geojson and CSV. As further specification, in the request header are included the metadata information about the dataset columns (API column name and data type specification). Using the same ID, the user can access the interactive API, available at the dev.socrata portal. All requests are processed by the system. In case of system overload, however, some requests may be rejected if not initiated using an authentication token. So, using an authentication token is strictly suggested. Apart from downloading datasets via the APIs, it is possible to use queries to select a set of data, using SQL (Socrata Query Language)[5].

The portal also provides the Maps section for geo visualisation and the Graph generator section for data visualisation. On top of accessing single datasets, users can also consult the Data lens section: this allows one to interactively analyse multiple datasets by using either graphs or maps. Lastly, users can use the Stories section to read articles and papers focused on a particular domain or topic, showing relevant datasets and some topic-specific data visualisation, maps and KPIs (Key Performance Indicators).

Figure 5 shows the dataset online visualisation in tabular format. On the right top corner there is a menu for data export: API access and various download modalities.



Dati sensori meteo

Dati rilevati da sensori in campo nel mese in corso.Il posizionamento dei sensori è disponibile qui: https://www.dati.lombardia.it/d/nf78-nj6b. NB: l'orario del dato è "ora solare"

Scarica una copia di questo dataset in formato statico

IdSensore	Data	Valore	IdOperatore	Stato
19295	09/05/2022 11:50:00 AM	0	4	VA
19300	09/05/2022 11:50:00 AM	1,9	3	VA
19300	09/05/2022 11:50:00 AM	0,8	1	VA
19301	09/05/2022 11:50:00 AM	337,6	1	VA
19301	09/05/2022 11:50:00 AM	314,2	3	VA
19307	09/05/2022 11:50:00 AM	14,5	1	VA
19308	09/05/2022 11:50:00 AM	62	1	VA
19309	09/05/2022 11:50:00 AM	0	4	VA
19333	09/05/2022 11:50:00 AM	16,6	1	VA
19334	09/05/2022 11:50:00 AM	49,9	1	VA
19335	09/05/2022 11:50:00 AM	0	4	VA
19342	09/05/2022 11:50:00 AM	15,2	1	VA
19343	09/05/2022 11:50:00 AM	73	1	VA
19344	09/05/2022 11:50:00 AM	0	4	VA
19345	09/05/2022 11:50:00 AM	24,6	1	VA
19346	09/05/2022 11:50:00 AM	46,1	1	VA
19347	09/05/2022 11:50:00 AM	0	4	VA
19354	09/05/2022 11:50:00 AM	22	1	VA
19355	09/05/2022 11:50:00 AM	52,4	1	VA
19356	09/05/2022 11:50:00 AM	0	4	VA
19363	09/05/2022 11:50:00 AM	20	1	VA
19364	09/05/2022 11:50:00 AM	59	1	VA
19365	09/05/2022 11:50:00 AM	0	4	VA

Mostra righe 1-100 di 1.403.638

Figure 5: Example of dataset online visualisation.

[5] Further information about this is accessible at this link and queries can be built interactively for each dataset.

2.2 ISPRA open data system

ISPRA (*Italian Institute for Environmental Protection and Research*) intends to adapt and develop its organisation according to the principles of transparency, efficiency, accountability of public administration and active participation of citizens, as required by European, national and regional legislation, identifying Open Data[6] as one of the concrete cornerstones of Open Government.

The public data contained in ISPRA's databases, produced or acquired in the performance of its institutional functions, unless otherwise specified are the property of the community, which has the right to access and re-use them: ISPRA makes public data held in its databases available through a specific section of the SINA (*National Environmental Information System*) in the context of SNPA (*National System for Environmental Protection*)[7].

Currently, almost all the databases in ISPRA (more than 200, including those related to EU/International projects) are published in Open Data, spanning from Air quality and emissions, Nature Conservation and Biodiversity, Soil and natural hazards, Sea and Coastal and other environmental domains.

The ISPRA Linked Open Data Platform is the component of the ISPRA Open Data system that was developed to respond to Linked Data design principles[8] for sharing machine-readable interlinked open data on the Web, thus fulfilling the FAIR (*findability*, *accessibility*, *interoperability* and *reusability*) data principles[9].

2.2.1 Data processing workflow

The linked open data processing workflow at ISPRA is reported in Figure 6 and described below. It is included also in the deployment diagram illustrated in Figure 7. The whole procedure can be divided into three phases: data extraction, preparation and transformation, and publication.

[6] Following the development of the "Open Data" at an international level and the transposition into Italian legislation of the relevant EU directives, all Public Administrations (PAs), and therefore ISPRA as well, should comply with the new regulatory obligations. To this end, the Italian Digital Agenda, a provision issued by the Italian Government aiming at fostering economic recovery in our country, envisaged its open data initiative for the PAs and assigned to AgID – Agency for Digital Italy – the task of coordinating a National Working Group to draw up a "draft" to which all PAs should refer for the publication of their data by virtue of article 52 of Legislative Decree, no. 82 of 7 March 2005 as amended by article 9 of Decree Law no. 179/2012: in this implementation, ISPRA also took into account the rules of the *Directive 2007/2/EC* of the European Parliament and of the Council of 2007, March 14th that facilitates public access to spatial information across Europe.

[7] https://www.isprambiente.gov.it/en/archive/news-and-other-events/ispra-news/2016/07/national-system-for-environmental-protection-snpa?set_language=en.

[8] https://www.w3.org/DesignIssues/LinkedData.html.

[9] https://www.go-fair.org/fair-principles/.

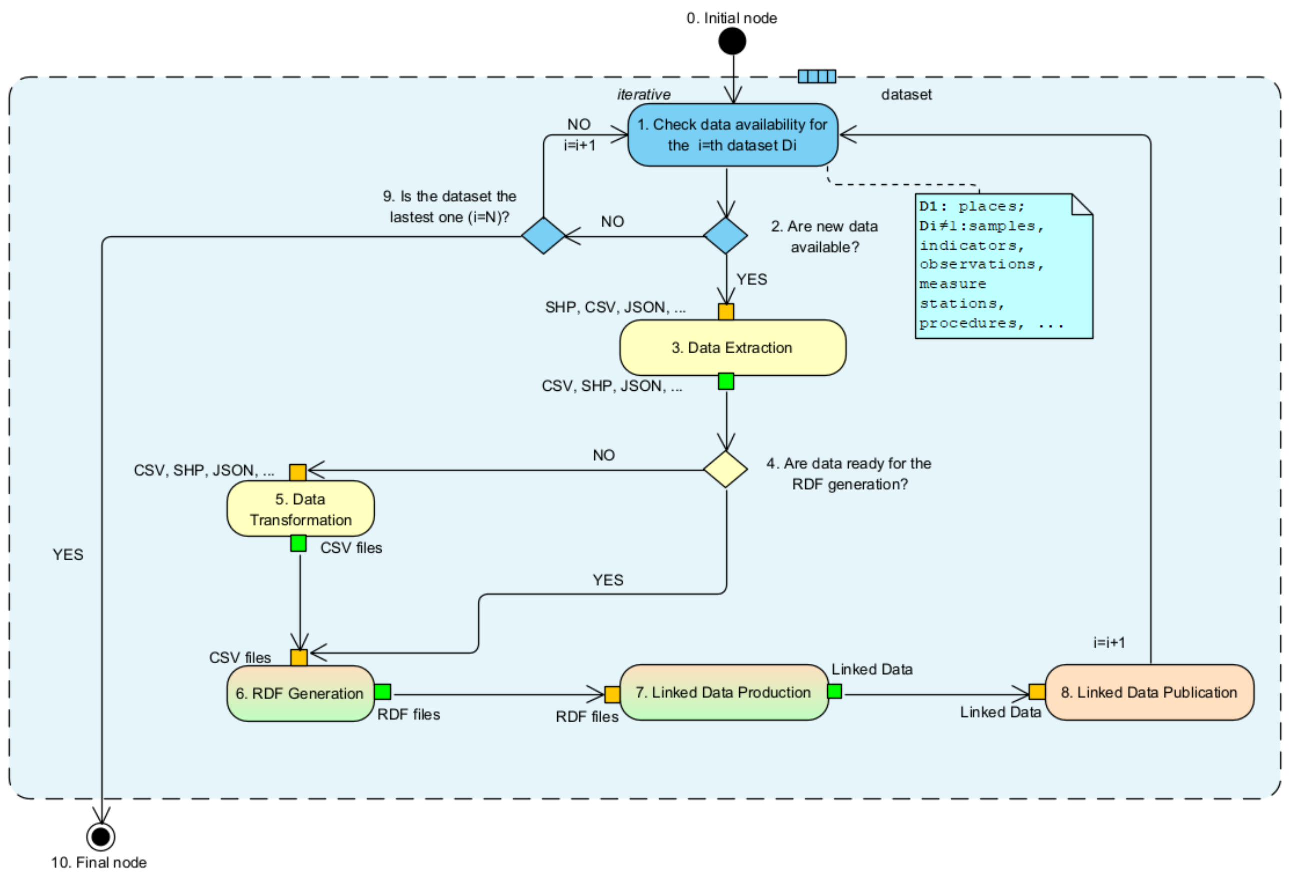

0. Initial node
iterative
dataset
1. Check data availability for the i=th dataset Di
NO
i=i+1
9. Is the dataset the lastest one (i=N)?
NO
2. Are new data available?
YES
D1: places; Di≠1:samples, indicators, observations, measure stations, procedures, ...
SHP, CSV, JSON, ...
3. Data Extraction
CSV, SHP, JSON, ...
NO
4. Are data ready for the RDF generation?
CSV, SHP, JSON, ...
5. Data Transformation
CSV files
YES
YES
CSV files
6. RDF Generation
RDF files
RDF files
7. Linked Data Production
Linked Data
Linked Data
8. Linked Data Publication
i=i+1
10. Final node

Figure 6: Linked Open Data production workflow at ISPRA.

Data Extraction and Transformation

Data to be published comes from different sources (e.g. databases, GIS layers, XML, JSON, NetCDF, GRID). This data could be already pre-processed in a previous stage, according to a set of procedures described in deliverable "Data pre-processing is finalized". It is important to extract these data directly from the original source and make them useful for a good RDF generation. "Data Extraction" (step number *3* in Figure 6) consists in a set of tools performed automatically in Python and/or R[10] or through database queries that select data to be published and generate the input file for the following steps. The automatic tools generally perform db queries, field selection within a Shapefile, download of measurement collections and metadata information by monitoring stations and generate CSV files, which contain all relevant information to be published.

As output, this step produces alternately:

- Input files (CSV format) for the "RDF Generation" step (number *6* in Figure 6)
- Input files for the "Data Transformation" step (number *5* in Figure 6)

"Data Transformation" step is performed only for specific extracted data that needs further tuning up to obtain the input (CSV) file for the "RDF Generation" step. As an example, this step allows to convert data

[10] R open source software (https://cran.r-project.org/)

from the original format to text (CSV) file and the "decoding" of all information to bring them to the UTF-8 standard encoding and avoid string interpretation errors.

The output CSV files are typically separated by year, since most of the indicators that ISPRA provides are calculated on a per-year basis. Datasets currently managed in Linked Open Data by ISPRA include: The National Tide Gauge Network (*RMN*), The National Data Buoy Network (*RON*), land consumption indicators, urban area indicators and The Repertory of Mitigation Measures for National Soil Protection (*ReNDiS*). Cross-linked to all these datasets is the places dataset, which with regard to administrative limits is based on ISTAT (National Institute of Statistics) data.

RDF Generation and Linked Data Production[11]

The CSV files are placed into a server which features a Python installation (*Triplifier Server*). A series of Python modules, specifically designed for this purpose, is then used to transform the input data stored in the CSV files into RDF data format, save triples on compressed text files and copy these files onto a second server (*SPARQL Server "Publisher"*, for RDF files bulk load into *Virtuoso*[12]). The Python modules are based on the *pyRML*[13] tool, which is based on the python *RDFLib*[14] module and on the RML[15] language for the tabular data mapping into RDF format. RML is a descriptive language expressed in RDF and allows to represent source data as RDF.

For some datasets (such as real time data: tide gauge and wave buoys networks) the Python modules provide the metadata only. Data recorded by the networks (*Real Time Nodes*) are loaded (*Triplifier Server* and *Web Repository*), in the form of periodical CSV files, using specific scripts performed in R open-source software, useful for data validation (already described in deliverable "Data pre-processing is finalized") and data collection.

Data Publication[16]

This parallel server features a *OpenLink Virtuoso* installation[17], which is used to load all triples in the triplestore. It acts as a master (*SPARQL Server "Publisher"*) server and is accessible via the internal network only. Triples stored in the master server are immediately mirrored into a slave (*SPARQL Server "Subscriber"*) server, which has read-only permissions and is also accessible from outside the internal network. This structure allows to protect the original data and to unload the Virtuoso server from external users' queries.

[11] This step involves the following steps in Figure 6: steps *6* and *7,* and the following component found in Figure 7: *Triplifier Server*.
[12] https://virtuoso.openlinksw.com/., *OpenLink Virtuoso Universal Server Enterprise Edition 8.x.*
[13] https://github.com/anuzzolese/pyrml.
[14] https://rdflib.readthedocs.io/en/stable/.
[15] https://rml.io/specs/rml/.
[16] This phase primarily involves the *step 8* in Figure 6 and the following components found in Figure 7: *SPARQL Server "Publisher"* and *SPARQL Server "Subscriber"*.
[17] https://virtuoso.openlinksw.com/, *OpenLink Virtuoso Universal Server Enterprise Edition 8.x.*

2.2.2 ISPRA Linked Open Data Platform Overview

Figure 7 shows the architecture of the ISPRA Linked Open Data Platform, followed by a functional description of its components.

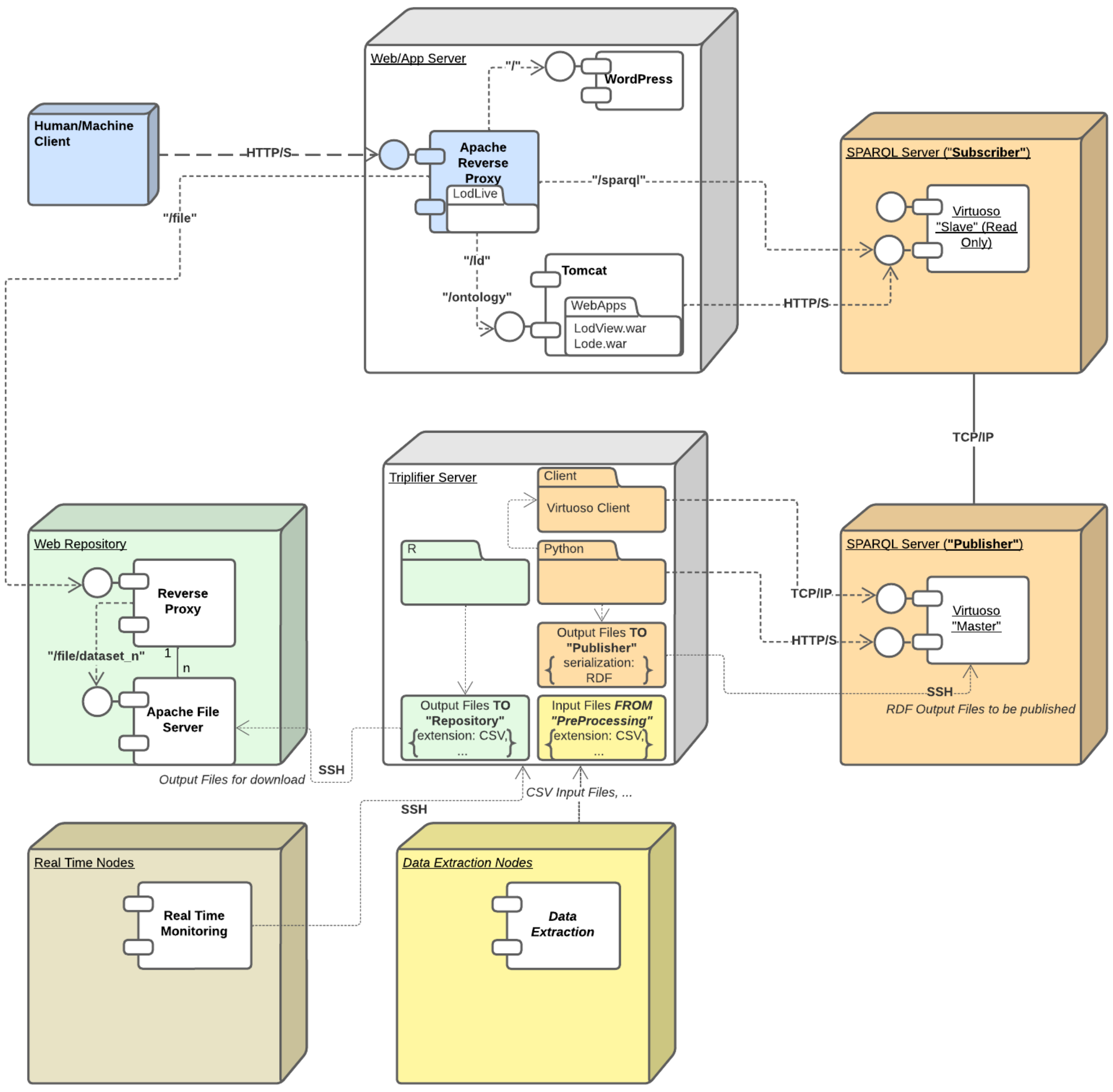


Figure 7: Deployment diagram for the ISPRA Linked Open Data Platform.

Apache Reverse Proxy (*Web/App Server*): it is the Single Infrastructure Access Point (SIAP) access node for the Client HTTP and HTTPS requests, including SPARQL endpoint queries, website access, semantic data navigation through domain applications (e.g. *LodView*, *LodLive*, etc.) and access to files and resources supporting LODs. The SIAP does not directly provide any of the above-mentioned functionalities, but it's meant to forward HTTP/HTTPS requests to the machines dedicated to the specific functionalities. The requests forwarding exploits a reverse proxy mechanism, enabled by an Apache web server (*Apache*

Reverse Proxy). More in detail, the reverse proxy mechanism uses forwarding rules based on the request path and could be potentially enabled to guarantee Cross-Origin Resource Sharing (CORS) requests. In Figure 1, this is exemplified by the directional connectors labelled with the forwarding-relevant part of the request path. For example, HTTP/HTTPS requests containing the parameter .../sparql/... in the request path will be forwarded to the machine where the triple store (*SPARQL Server "Subscriber"*) is located and which provides the data query interface via the SPARQL endpoint. The logic of the other connectors with the SIAP as source node is similar.

Web storage (*Web Repository*): this is the system devoted to download resources that can be linked to LODs but are not directly dereferencable, according to Semantic Web standards[18]. In particular, the resource describing access to this data is published in LOD. The related access link may allow to:

- download RDF datasets;
- download observation data from measurement stations (e.g. monthly CSV files with near real-time data measured by stations belonging to the The National Tide Gauge (*RMN*) or The National Data Buoy (*RON*) networks).

In general (not represented in this diagram) this link may direct to further services that allow access to the data (e.g. OGC APIs[19]).

Real Time Nodes (*Real Time Monitoring*): these are the nodes that provide external access to the data produced in near real-time, e.g. by the *RMN* and *RON* networks managed by ISPRA. The separation with respect to the SIAP is dictated by isolation requirements, necessary both to guarantee a higher level of security and to avoid the overload of the dedicated LOD servers. The *Real Time Nodes* are designed to accept requests via HTTP and HTTPS protocol. Communication between the nodes and the servers dedicated to LODs typically takes place via access to the *Triplifier Server*, in order to process the data to be published in LOD. It may also happen that the *Web Repository* is accessed too, to associate the relevant non-RDF resources.

Web Site (*WordPress*): website of the ISPRA LODs. It is implemented using the *WordPress* content management system and HTML, RDFa, CSS and Javascript technologies as a framework.

In addition to a general introduction about the Linked Open Data project in ISPRA *(Linked ISPRA*), the website consists of the following sections for each dataset published:

- *DATASETS*: here the individual dataset is described with example of SPARQL queries;

- *DATA ACCESS*: here it is possible to download the RDF graph associated with the dataset, navigate to the SPARQL endpoint and to a geographic viewer of the published data down to the specific individual resource;

[18] https://www.w3.org/DesignIssues/LinkedData.html: Tim Berners-Lee proposed 4 principles to publish Linked Data: *1) Use URIs as names for things; 2) Use HTTP URIs so people can look up those names; 3) When someone looks up a URI, provide useful information using the standards; 4) Include links to other things, so people can discover more.*
So to be "dereferencable", a URI should be usable to retrieve a useful representation of the resource it identifies, using standard technologies as RDF and SPARQL.
[19] https://ogcapi.ogc.org/features/.

- *ONTOLOGIES AND THESAURI*: here it is possible to view, navigate and perform RDF download of ontology and thesauri published by ISPRA. **Semantic Browser**: this is the component that hosts the applications for LODs navigation and interaction. It consists of *LodView*[20] (web app on the application server *Tomcat*) and *LodLive*[21] (Javascript on front end *Apache Reverse Proxy*).

LodView proposes a way to experience the *web of data*: in particular it offers the right representation of the information requested by users (by providing for the dereferencing of IRI) being able to understand whether a human or a machine is accessing a resource[22].

LodLive too uses Linked Data standards (RDF, SPARQL) to browse RDF resources, but it is more focused toward a visual representation of graphs related to them.

Data extractor *(Data Extraction Nodes)*: this component is aimed at extracting data, already processed by the previous pre-processing phase, in order to prepare them for proper publication in Linked Open Data. At this stage CSV spreadsheets from relational databases, Shapefiles and XML, for example, will be generated.

Triplifier and data source manager (T&DSM) (*Triplifier Server*): this is the machine where the LOD production scripts run, implemented in Python and R language. The production of LODs is based on the conversion of non-RDF data sources into RDF. Non-RDF files are derived from *Data extractor*. The *Triplifier Server* can connect to the *SPARQL Server "Publisher"* interfaces to publish data in LOD format (via TCP/IP iSQL/ODBC port or HTTP/HTTPS web interface), and to the *Web Repository* to generate any associated resources, like monthly CSV files with near real-time data measured by stations belonging to *RMN* or *RON* networks.

Triple store & SPARQL endpoint (*SPARQL Server "Subscriber"*): it is the machine where the triple store is installed. It is based on *OpenLink Virtuoso*[23], configured as a "Subscriber" node in read only mode. It responds to public HTTP/HTTPS requests via reverse proxy, and is aligned in real time with the data contained in the analogous "Publisher" node (*SPARQL Server "Publisher"*), which is not accessible. Besides providing dedicated technologies for storing RDF triples, the triple store exposes the endpoint that enables SPARQL requests over HTTP(S). The SPARQL language version supported is 1.1 and also includes write requests (UPDATE) upon authentication. In addition, the *Virtuoso*-based SPARQL endpoint is configured to natively support the GeoSPARQL extension, which is a standard for representing and querying geographic Linked Data.

[20] https://lodview.it; https://github.com/LodLive/LodView.
[21] http://lodlive.it/; https://github.com/LodLive/LodLive.
[22] 1) a user requests an RDF resource, an IRI, published on the web; 2) LodView analyses the content of the user's request and provides an adequate representation of the resource (If an HTML representation was requested for example, the HTML representation provides information on the available properties, on the connected objects, on the geographical coordinates and on the images present in the resource); 3) if the RDF resource contains links to other resources published on *the data web* LodView explores the connections and shows the information, demonstrating the true value of Linked Open Data (5 stars).
[23] https://virtuoso.openlinksw.com/, *OpenLink Virtuoso Universal Server Enterprise Edition 8.x*

3 The Linked Open Data production process

Figure 8 shows a high-level overview of the linked open data production process we envisage for the WHOW datasets. Heterogeneous data sources are available from data providers (ISPRA and Aria SpA) and external providers (e.g. Eionet, Copernicus). These are acquired and pre-processed before their transformation into RDF according to the WHOW ontology network, a network of interconnected ontologies that we are currently developing in order to provide a harmonised semantic layer for all datasets considered in the three WHOW use cases. The pre-processing activities are all those described in Deliverable 3.1 - "Data pre-processing is finalised" [9].

Once the datasets are available in the WHOW linked open data production pipeline, they undergo a triplification process that consists in transforming them in RDF datasets (data under the form of triples `subject- predicate-object`) possibly linked to other RDF datasets available in the so-called Web of Data; that is, the Web of interlinked machine-readable data (see Figure 8). This contributes to the creation of the WHOW knowledge graph. The triplification together with a phase of data quality assessment, in turn carried out according to all the principles that have been extensively described in Deliverables 5.1 and 3.1, form an iterative process that allows us to incrementally improve the resulting knowledge graph (linked open data). This latter is then made available in data catalogues in order to augment its findability and for the re-use by anyone for any purpose (e.g., in order to construct any types of data views and/or analytics with it).

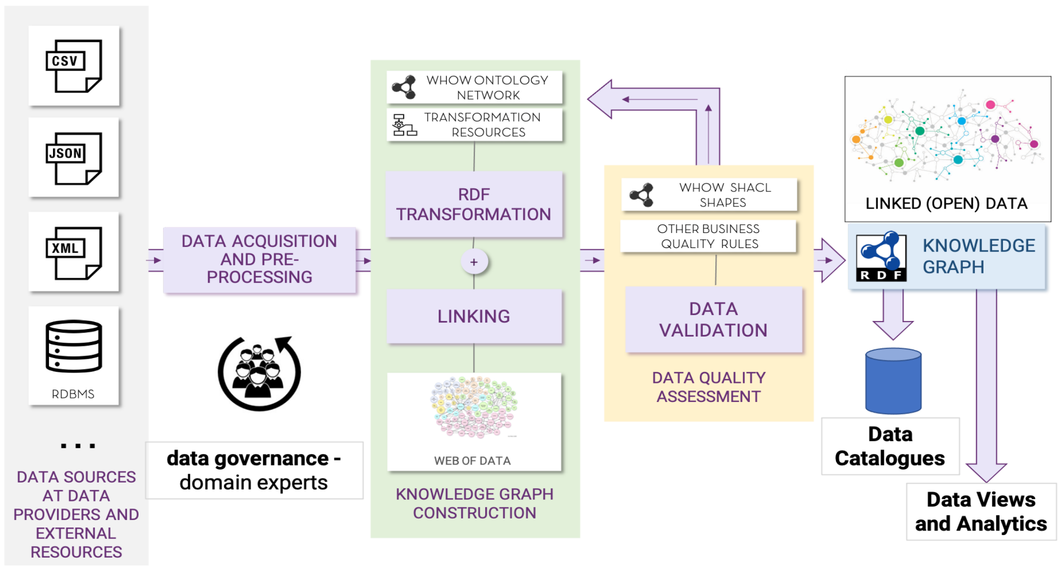
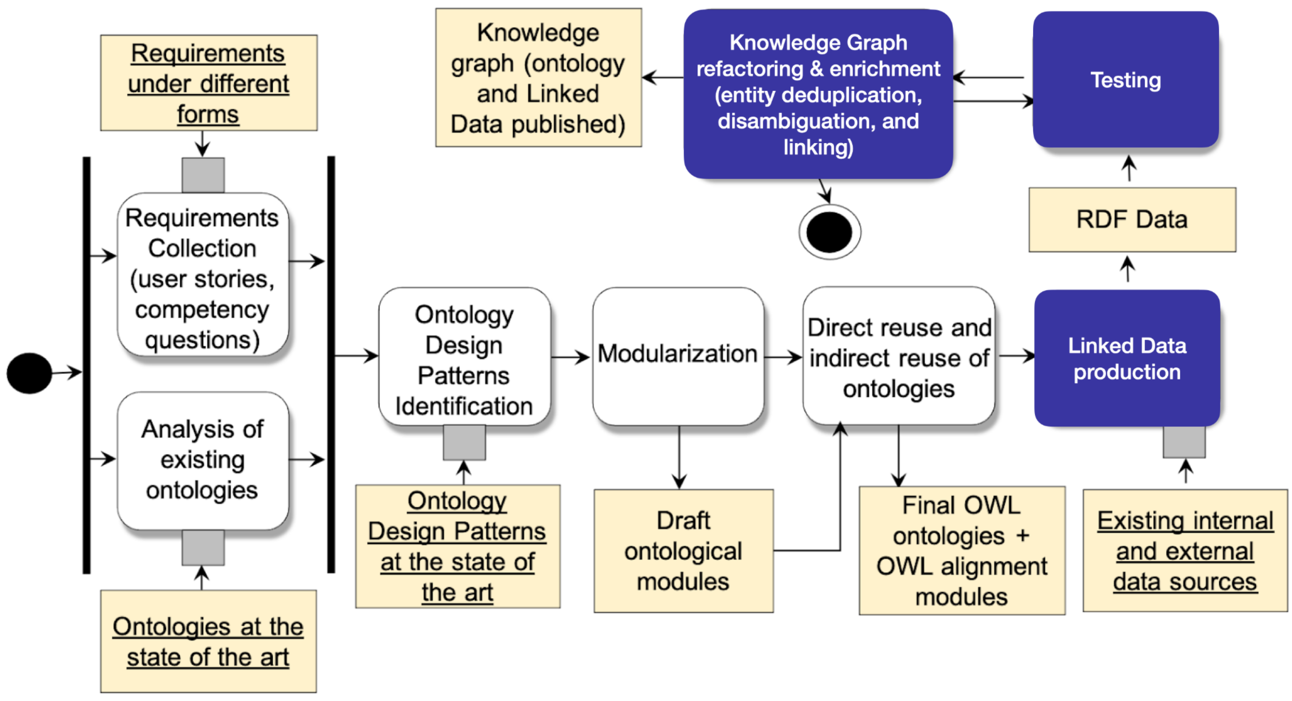


Figure 8: High-level view of the linked open data production process.

Figure 9 illustrates the linked open data production process from a broader perspective. In fact, the production of linked open data in WHOW is part of a larger workflow that includes different complementary and preparatory steps, such as, the definition of ontological requirements, modular ontology design, linked data production, and testing. Strictly speaking, this deliverable is focused on the activities highlighted by the violete boxes in Figure 9, i.e. (i) Linked Data production; (ii) Testing; and (iii) Knowledge Graph refactoring & enrichment (entity deduplication, disambiguation, and linking).



Requirements under different forms
Requirements Collection (user stories, competency questions)
Analysis of existing ontologies
Ontologies at the state of the art
Ontology Design Patterns Identification
Ontology Design Patterns at the state of the art
Modularization
Draft ontological modules
Direct reuse and indirect reuse of ontologies
Final OWL ontologies + OWL alignment modules
Linked Data production
Existing internal and external data sources
RDF Data
Testing
Knowledge Graph refactoring & enrichment (entity deduplication, disambiguation, and linking)
Knowledge graph (ontology and Linked Data published)

Figure 9: UML activity diagram for the knowledge graph production process.

By Linked Data production we mean the generation of RDF data from non-RDF sources (e.g. CSV, relational databases, spreadsheets, etc.) by applying the linked data paradigm and FAIR principles. This process is detailed by the UML activity diagram depicted in Figure 10. This figure adds a more granular representation to the general diagram depicted in Figure 9. In Figure 10, different colours are used for identifying different activities. Namely:

- the actions in white, i.e. action *"1. Data preparation and cleansing"*, are those we describe in the deliverable 3.1 associated with milestone 4 [9];
- light green is used for highlighting the actions aimed at producing RDF triples from non-RDF data, i.e. triplification;
- light orange is used for highlighting the actions aimed at validating produced RDF graphs;
- blue highlights the actions aimed at enriching the RDF graph with new links either internal or towards external linked open datasets;
- grey highlights the data publication step.

The yellow boxes pinned to actions identify the inputs required by actions themselves. On the contrary, green boxes pinned to actions identify the outputs produced by those actions. In the next subsections we provide more details for each of the actions presented in Figure 10. The all process will be implemented by means of an orchestration management platform for data engineering pipelines. We plan to use for this Apache Airflow[24].

[24] https://airflow.apache.org/

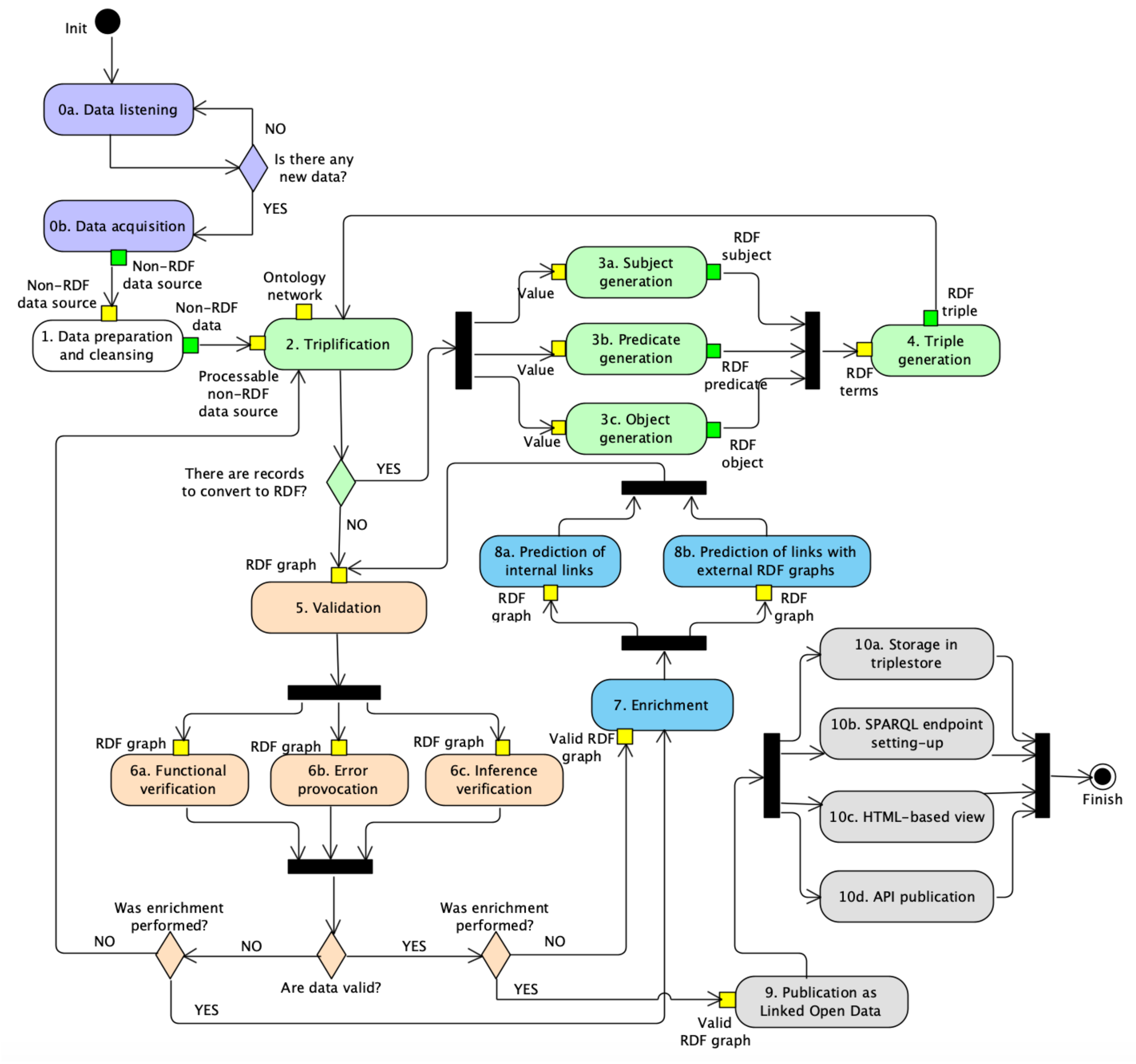


Figure 10: UML activity diagram for the linked open data production process.

3.1 Data acquisition

The data acquisition step allows WHOW to get external non-RDF open data and start the transformation process to Linked Open Data (LOD) represented as RDF. We assume an action of our workflow focused on listening to possible new external data that are fed to the transformation process in order to generate (LOD). The listeners will be part of the orchestration management platform, e.g. Apache Airflow. The action meant for listening to events happening in the system is *"0a. Data listening"*. Such an action listens to "new data available" events and notifies the software components implementing the action labelled as *"0b. Data acquisition"* to acquire those data. The workflow depicted in Figure 10 for data acquisition assumes that the data listening action is kept active till new data to process are detected. Then, the acquisition is performed before starting the data preparation, RDFizsation, validation, and publication. We assume that the data listening action is always running, i.e. it notifies the *"0b. Data acquisition"* action and keeps on listening to possible events.

The data acquisition part of our workflow, which is coulored in purple in Figure 10, addresses the high level requirement HLR-01 (Data provision) as defined in [10]. More specifically, it implements the component *OpenDataAcquirer* of the WHOW architecture designed in [10]. Accordingly, the data acquisition actions addresses the function requirement FR-03[25].

3.2 Triplification

Despite knowledge graphs being necessary to manage, organise and integrate data in an effective way, most data is still not available in the form of knowledge graphs. This is the case of WHOW data as well: it comes in different structures, heterogeneous formats and can be accessed via heterogeneous interfaces. The triplification is the generation of RDF statements, i.e. triples, from non-RDF sources. An RDF statement is composed of a subject, a predicate, and an object. Accordingly, the triplication can be divided into three sub-actions, that is, an action for generating the subject of an RDF statement, another for generating the predicate, and, finally, an action for generating the object. The output of the triplification is a set or RDF triples that come from the processing of an input non-RDF data source. This set of RDF triples is referred to as an RDF graph. The actions highlighted in light green implements the triplifier component that is described in the WHOW Deliverable 4.1 [10]. Accordingly, they address the requirements FR-0401, FR-06, FR-07, NFR-04 as described in [10].

Mapping-based approaches provide a solution to the implementation of triplifiers by filling the gap of generating knowledge graphs through rules applied to multiple data chunks that follow the same structure patterns. In other words, they are able to translate non-RDF data into RDF.

Due to the heterogeneous nature of the data, an increasing number of mapping languages have been proposed over time, and some of them have been listed in the Architecture Deliverable 4.1 [10] as viable options for the Triplifier component, as shown in Table 2.

Table 2: Mapping languages taken into consideration for the WHOW Triplifier software component.

Component	**Requirements**	**Tool**
Triplifier	**FR-0401, FR-06, FR-07, NFR-04**	**RML[26], RMLMapper[27], RMLStreamer[28], R2RML[29], pyRML[30], OpenLink Virtuoso**

[25] FR-03: *"The platform shall support the transformation of data managed through relational database management systems (RDBMSs) and accessible via SQL".*

[26] https://rml.io/specs/rml/.

[27] https://github.com/RMLio/rmlmapper-java.

[28] https://github.com/RMLio/RMLStreamer.

[29] https://www.w3.org/TR/r2rml/.

[30] https://github.com/anuzzolese/pyrml.

		(Sponger)[31] **SDM-RDFizer**[32] **SPARQL anything**[33] **SPARQL Generate**[34] **ShExML**[35]

These can be further divided into three categories: RML [11] and R2RML-based (RMLMapper, RMLStreamer, pyRML, SDM-RDFizer), SPARQL-based (SPARQL anything [2], SPARQL generate [12], OpenLink Virtuoso) and ShEX-based (ShExML [13]). According to [6], some languages, inside these categories, are similar to each other, and others can be related even if belonging to different groups (ShExML and RML, for instance). The SPARQL-based group is instead considered isolated from the other groups due to the potentialities of relying on SPARQL.

In this section, we assess one representative of each of the three groups (RML, SPARQL anything, ShExML.) with the addition of the mapping language Ontop[36]. Specifically, RML is "a superset of the W3C-standardised mapping language [R2RML], aiming at extending its applicability and broadening its scope for supporting the mapping of data available in structured formats other than relational databases (e.g., CSV, JSON, XML, etc.)"[37]. SPARQL-anything [2] uses SPARQL, which is the standard query language for the Semantic Web; It is based on the Facade-X abstraction that in turn uses a subset of RDF as a general approach to represent the source content *as-it-is* but in RDF. Ontop relies on SPARQL and R2RML mappings[38]; finally, ShExML is a language based on the Shape Expressions (ShEx) language which is not a standard as of today[39].

This assessment takes into consideration [4]'s framework as a set of requirements for a comparison between mapping languages, which are distinguished into functional, non-functional, and data source support characteristics, as shown in Table 3. These requirements have been extended for the sake of completeness. The requirements are:

- NF1: *Easy to use by semantic web experts*: Whether the mapping language is easy to use by Semantic Web experts to describe the generation process or not; that is, a syntax that is either "familiar" (SPARQL-like) or "human-readable" to write rules;
- NF2: *Based on semantic web standards*: Whether the mapping language integrates with a typical semantic web engineering workflow" [12] or not, i.e., if it is related to existing standards;

[31] http://vos.openlinksw.com/owiki/wiki/VOS/VirtSponger.

[32] https://github.com/SDM-TIB/SDM-RDFizer.

[33] https://github.com/SPARQL-Anything/sparql.anything.

[34] https://ci.mines-stetienne.fr/sparql-generate/.

[35] https://github.com/herminiogg/ShExML.

[36] https://www.w3.org/2001/sw/wiki/Ontop.
[37] See https://rml.io/specs/rml/.
[38] See https://ontop-vkg.org/guide/#versions.
[39] See https://shex.io/shex-semantics/.

- NF3: *Fully covering the generation process*: Whether the mapping language allows for serialisation of the generation process for further reuse or not: whether the generation description is fully covered by the mapping language, or certain parts are hard-coded;
- NF4: *Easy to use functions*: whether the functions are very verbose and hard to use or easy to use;
- NF5: *Well-known in the knowledge graph community with an extensive documentation and examples of uses;*
- DS1: Support heterogeneous data formats and heterogeneous data sources;
- F1: *Supporting general mapping functionalities*: Whether general mapping functionalities are provided or not, namely, specifying "M:N relationships", "literal to IRI", "vocabulary reuse", "data types", "named graphs", and "blank nodes". These functionalities include (i) generating subject, predicate, object, and (optionally) graph resources, (ii) joining data, and (iii) specifying the used ontology and data types;
- F2: *Extensible*: Whether additional functionalities can be added or not, to allow data to be manipulated and transformed;
- F3: *Supporting data manipulation functions*: Whether core functionalities involve data manipulation functions;
- F4: *Supporting Virtual Knowledge Graphs*: Whether the language supports Virtual Knowledge Graphs or not;
- F5: *Supporting nested hierarchies*: Whether nested data records can be joined or not, to map data elements from rows as well as structured values. This characteristic relates to whether the mapping language itself can handle hierarchical data structures or not;
- F6: *Supporting collections and lists*: Whether the mapping language allows for generating hierarchical values in the form of RDF collections or containers or not.

Table 3: Comparative analysis of mapping languages.

Language	NF1	NF2	NF3	NF4	NF5	DS1	F1	F2	F3	F4	F5	F6
RML	✓	✓	✓	✘	✓	CSV, TSV, XML, JSON, HTML, Bin., RDBMS, RDF, Text	✓	✓	✓	✘	✓	✘ (but some implementation engines support it
SPARQL anything	✓	✓	✓	✘	✘ (start being considered only recently)	JSON, CSV, TSV, HTML, Bin., XML, Text, Embed, Meta., Spread.	✓	✓	✓	✓	✓	✓
Ontop	✓	✓	✓	✘	✓	RDBMS	✓	✓	✓	✓	✓	✓
ShExML	✓	✘	✓	✘	✘	JSON, CSV, TSV, XML, RDBMS	✓	✘	✓	✘	✘	✓

As seen in Table 3, NF1 is addressed by all mapping languages, as it implies a syntax that is either already "familiar" (SPARQ-like, RDF) or "human-readable". The second requirement of the framework i.e., NF2, is instead acceptable for all tools taken into consideration, as noted in their official documentation, with the exception of ShExML. Furthermore, all languages comply with NF3, and propose a diverse set of data source supports.

As far as the WHOW dataset formats are concerned, both RML and SPARQL anything provide the necessary support, though there are datasets in XLSX that are not considered by any of the mapping languages listed in Table 3. However, XLSX can be easily transformed into CSV so that RML, SPARQL anything and ShExML can understand it.

As for functional requirements, all languages support general mapping functionalities and are extensible, besides ShExML that has not been extended yet. The last two functional requirements; that is, hierarchical data structures handling and support of collections and lists, are identified by the W3C community group for Knowledge Graph Construction among the nine challenges for declarative mapping languages[40]. Each challenge provides a high level problem statement of a feature that current mapping languages cannot always successfully handle. There is also a set of input files and the expected output graph for each challenge. Of these nine challenges, five are related to nested iterations and references returning multiple values, namely: access-fields-outside iteration, generate-multiple-values, multivalue-references, process-multi value reference and RDF-collections. On this point, [3] proves these challenges can be handled with RML, and [4] adds that SPARQL-based mapping languages are assumed to support this, as they extend SPARQL; in the case of ShExML, it uses iterators to extract information from input data, which can be defined to handle nested input data [5].

Finally, another important requirement is the adoption in the knowledge graph community and the availability of documentation and working examples to use the language (requirement NF5 in Table 3). With this regard, RML is probably the most famous mapping language: it is widely adopted in the community and a number of RML engines other than the main one RMLMapper are available at the state of the art.

Final conclusion. Based on this analysis, we can conclude that for the purposes of the WHOW project, RML can be successfully used, although it currently does not provide full support for virtual knowledge graphs, useful when dealing with external data sources. In this sense, the use of RML can be integrated with the use of either SPARQL anything or Ontop that in contrast support such a functionality.

3.2.2 RML mapping example of a WHOW dataset

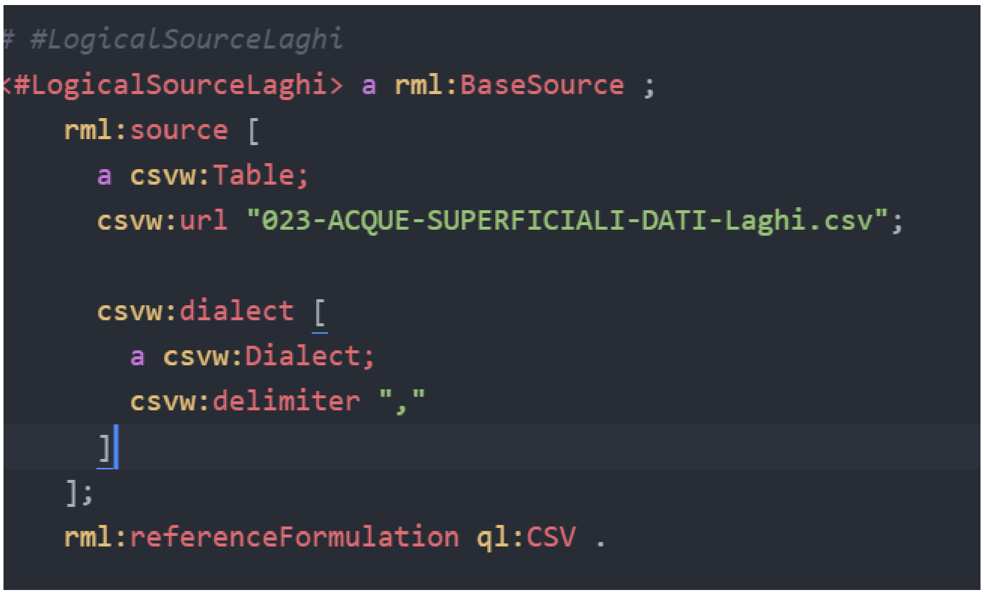
In this section, we present a first example of implementation of the triplifier component of the WHOW Linked Open Data Reference Architecture that exploits the RML mapping language and the RMLMapper[41] as an engine for constructing knowledge graphs.

When defining an RML script the first instruction to provide is the definition of the so-called Logical Source; that is, the original input dataset that has to be transformed in RDF. Figure 11 shows how this can be done

[40] See https://github.com/kg-construct/mapping-challenges/tree/main/challenges and the solutions https://github.com/RMLio/mapping-challenges-rml-fields/.

[41] https://github.com/RMLio/rmlmapper-java

with one dataset of Lombardy region that is related to the water monitoring of a specific type of surface water; i.e., lakes..

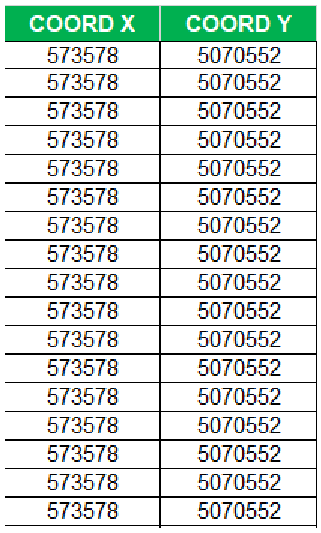


```
# #LogicalSourceLaghi
<#LogicalSourceLaghi> a rml:BaseSource ;
   rml:source [
     a csvw:Table;
     csvw:url "023-ACQUE-SUPERFICIALI-DATI-Laghi.csv";

     csvw:dialect [
       a csvw:Dialect;
       csvw:delimiter ","
     ]
   ];
   rml:referenceFormulation ql:CSV .
```

Figure 11: Mapping example of the logical source in RML.

The rest of the RML mapping aims to map geometry data using the ontology elements we developed to deal with geometries and geospatial data in general. The dataset, entitled *Analytical data on Lake Water Bodies* by ARPA Lombardia[42], has two columns related to geometry, as shown in Figure 12.

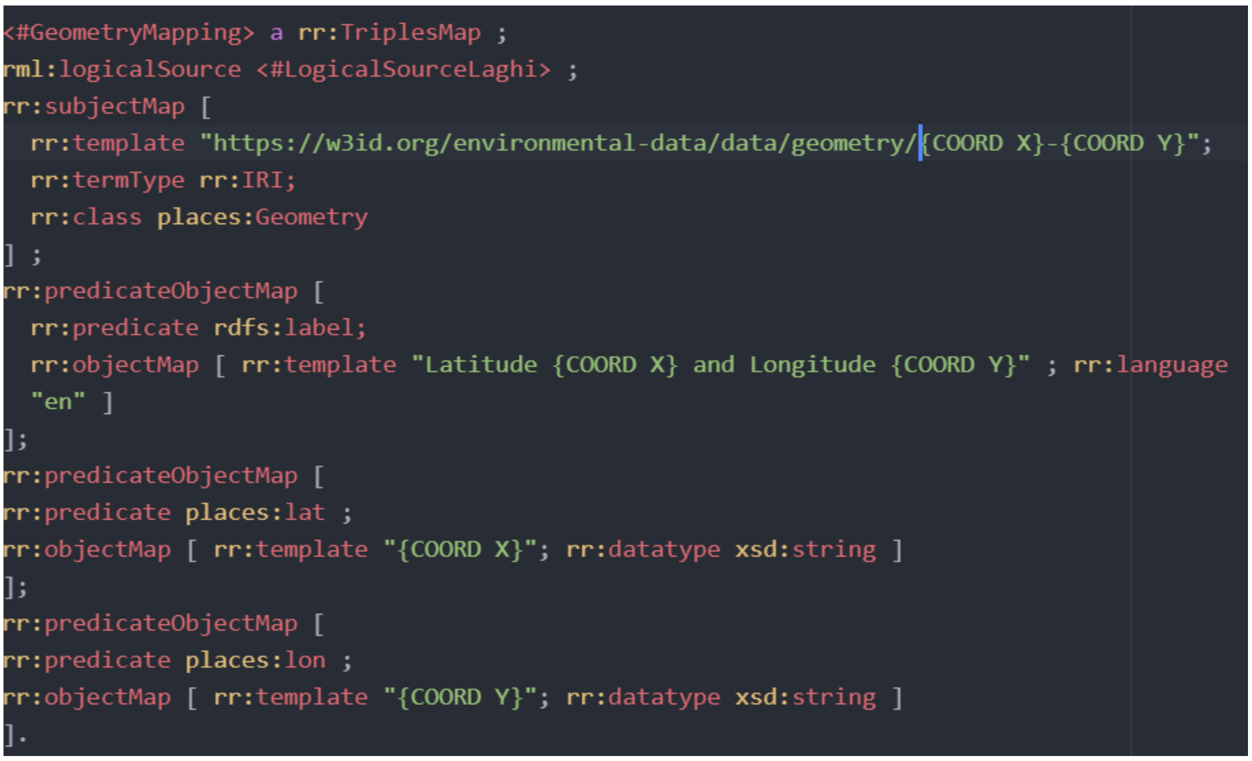


COORD X	COORD Y
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552
573578	5070552

Figure 12: "Coord X" and "Coord Y" - Geometry coordinates in the example dataset.

[42] The dataset is available at this link.

Figure 13 illustrates the mapping of the geometry coordinates found in the dataset of Figure 12 using specific ontology elements we foresee.



```
<#GeometryMapping> a rr:TriplesMap ;
rml:logicalSource <#LogicalSourceLaghi> ;
rr:subjectMap [
  rr:template "https://w3id.org/environmental-data/data/geometry/{COORD X}-{COORD Y}";
  rr:termType rr:IRI;
  rr:class places:Geometry
] ;
rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:template "Latitude {COORD X} and Longitude {COORD Y}" ; rr:language
  "en" ]
];
rr:predicateObjectMap [
rr:predicate places:lat ;
rr:objectMap [ rr:template "{COORD X}"; rr:datatype xsd:string ]
];
rr:predicateObjectMap [
rr:predicate places:lon ;
rr:objectMap [ rr:template "{COORD Y}"; rr:datatype xsd:string ]
].
```

Figure 13: Mapping of geometry coordinates example in RML.

In Figure 13, a specific `TripleMap` is created for the mapping of the geographical coordinates. A triple map specifies a rule for translating each row of a table to zero or more RDF triples. The logical source from which the triple map acts is the one defined in Figure 11. Since the coordinates are modelled using a class named Geometry, the first element to be created is the instance of the class Geometry that has its own IRI. This instance is the subject of the RDF triples (`rr:subjectMap` in Figure 13). We decided that, in order to guarantee that IRI is unique and persistent over time, both coordinates can be used in its creation.

Please note that, the IRI of Figure 13 follows the 10 rules for persistent URIs[43] for which it is formed as follows:

`https://{namespace/domain}/{type}/{concept}/{reference}`

`https://w3id.org/environmental-data/data/geometry/573578-5070552`

The `rr:template` illustrated in Figure 13 is the directive R2RML, which RML extends, to be used to create the IRI.

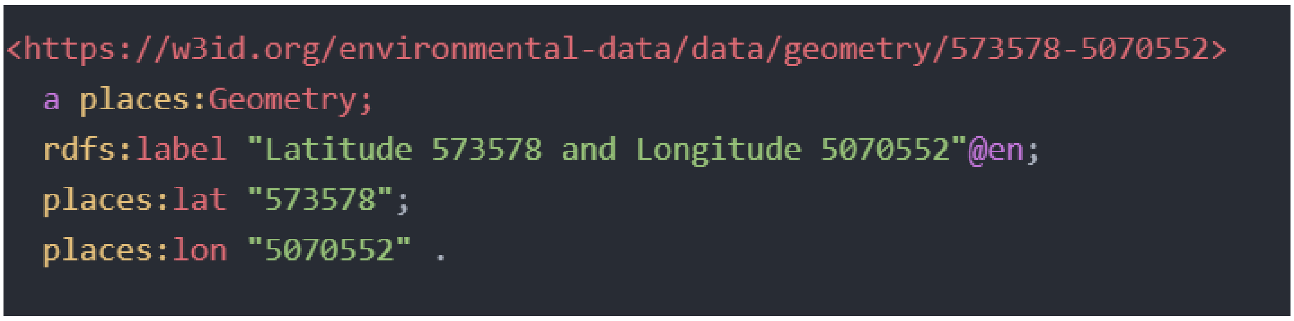
Once the subject is defined, the remaining part of the triple map are the predicates that link the subject to the various objects. In Figure 13, one of these predicates is `rdfs:label`, useful for human-readability of

43 https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/document/10-rules-persistent-uris

the produced RDF data, that can be navigated online through software that renders it in HTML. This predicate allows one to specify a human readable label to be associated with any entity of the application domain.

In addition to the label, other two important predicates are associated with the Geometry class. These are the actual coordinates expressed in latitude and longitude. In particular, as depicted in Figure 12 through the predicates `places:lat` and `places:ling` we can map the two columns of the dataset in Figure 12 above. These two predicates are data type properties defined in the ontology `places`..

The above RML script is then used as input for the RMLMapper engine that produces an output in RDF Turtle, one of the various RDF serialisations, for every row in the dataset, as shown in Figure 14 As it is possible to see, the original dataset is now mapped into RDF triples.



```
<https://w3id.org/environmental-data/data/geometry/573578-5070552>
  a places:Geometry;
  rdfs:label "Latitude 573578 and Longitude 5070552"@en;
  places:lat "573578";
  places:lon "5070552" .
```

Figure 14: RDF output for one row of the dataset.

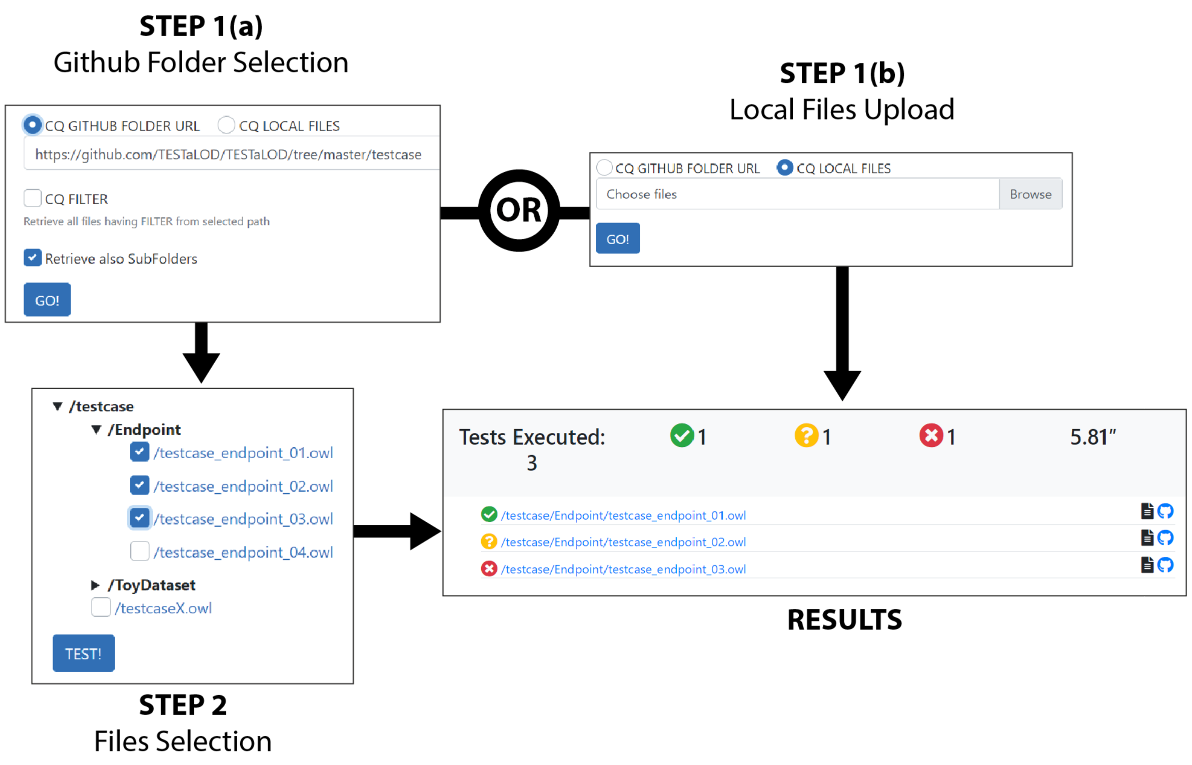
3.3 RDF graph validation

We design a solution to the RDF graph validation that relies on the framework envisioned by [14]. Namely, we assess the validity of an RDF graph along three different dimensions: (i) functional, (ii) logical, and (iii) structural. The functional dimension is related to the intended use of a given RDF knowledge graph (KG) and of its components, i.e. their function in a context. It is a core dimension for KG testing. In fact, it allows designers to assess the ability of a KG to address requirements and cover the domain. The logical dimension measures whether a KG can be successfully processed by a reasoner (inference engine, classifier, etc.). Finally, the structural dimension of a KG focuses on its topological properties measured by means of context-free metrics that leverage its graph-based representation. The functional and logical dimensions are utmost important for assessing the quality of a KG. Nevertheless, the analysis of the structural dimension provides insights on design choices by means of indicators that might suggest weaknesses and strengths. The logical dimension is addressed by running a reasoner (e.g. HermiT[44] [15]). The functional dimension is assessed by means of the competency questions (CQs), i.e. natural language questions that express the functional requirements of a KG. Hence, we focus on evaluating the appropriateness of a KG against its requirements intended as the ontological commitment expressed by means of CQs and domain constraints, i.e. functional dimension. User stories are translated into one or more CQs and general constraints during the design phase. To each CQ and to each constraint corresponds a unit test, which contributes, when run, to validate the KG. Thus, the core element of each unit test is either a CQ or a general constraint (e.g. disjointness axiom). Accordingly, this approach consists in testing whether the KG allows the conversion of a CQ, reflecting a requirement, to a SPARQL query. This step allows the detection of any missing knowledge or

[44] http://www.hermit-reasoner.com/

gap in the KG. The action *“6c. Inference verification”* focuses on checking the inferences over the KG, by comparing a set of expected inferences to the actual ones produced by an automatic reasoner. If such a reasoner does not materialise some of the expected inferences this means that an appropriate axiom or piece of knowledge is missing from the KG. Hence, the triplification process should be reworked. This third testing action, i.e. *“6b. Error provocation”* is intended to stress the knowledge graph by injecting inconsistent data that violate the requirements. The expected result is an inconsistency: if this is not detected by the reasoner, it means that the appropriate axioms or triples are missing in the KG. Hence, the triplification process should be reworked .All the actions depicted in Figure 10 for KG validation realises the *DataValidator* component as described in the WHOW deliverable 4.1 [10]. The DataValidator addresses the WHOW functional requirement *FR-11* (and all its sub requirements).

A possible software component we can rely on for implementing the KG validation is TESTaLOD [16]. TESTaLOD is a tool designed and implemented for testing KG modelled by means of the eXtreme Design (XD) [17] agile methodology or other test-driven methodologies. TESTaLOD is developed as a Web application[45] that provides a knowledge graph testing toolbox: as presented in Figure 15, it implements a two-step workflow, allowing the user to select and automatically run defined test cases aiming at verifying CQs. The test cases are OWL files, and are modelled by using the TestCase OWL meta model introduced in [18], thus containing: a Competency Question and its corresponding SPARQL query, the expected (correct) result and data sample. The test cases can be either retrieved from a GitHub repository or uploaded from a local file system. Once the tests have been au- tomatically executed, the expected result is compared to the actual result, and three possible outputs can be displayed to the user: successful, partially successful, unsuccessful.


STEP 1(a)
Github Folder Selection
CQ GITHUB FOLDER URL
CQ LOCAL FILES
https://github.com/TESTaLOD/TESTaLOD/tree/master/testcase
CQ FILTER
Retrieve all files having FILTER from selected path
Retrieve also SubFolders
GO!
OR
STEP 1(b)
Local Files Upload
CQ GITHUB FOLDER URL
CQ LOCAL FILES
Choose files
Browse
GO!
▼ /testcase
▼ /Endpoint
/testcase_endpoint_01.owl
/testcase_endpoint_02.owl
/testcase_endpoint_03.owl
/testcase_endpoint_04.owl
▶ /ToyDataset
/testcaseX.owl
TEST!
STEP 2
Files Selection
Tests Executed: 3
1
1
1
5.81”
/testcase/Endpoint/testcase_endpoint_01.owl
/testcase/Endpoint/testcase_endpoint_02.owl
/testcase/Endpoint/testcase_endpoint_03.owl
RESULTS

45 https://w3id.org/testalod

Figure 15: Workflow implemented by TESTaLOD.

3.4 RDF graph enrichment

The objective of RDF graph enrichment is twofold: (i) predicting new links between resources already described in the graph, i.e. *intra*-linking and (ii) generating alignments with resources available in external linked open datasets, i.e. *inter*-linking, for maximising knowledge interoperability and reusability. Intra-linking has been studied under the prism of knowledge graph link prediction, primarily by using machine learning and neural network architectures [19]. In this scenario an RDF graph is a collection of facts represented as nodes and the edges linking these nodes, respectively. Each fact is represented as a triple *(h,r, t)*, where *h* is a head entity, *t* is a tail entity and *r* is the relation between them. The goal of link prediction is to predict missing information (links or relations) between the entities in KGs. Figure 16 shows an illustrated example of link prediction.

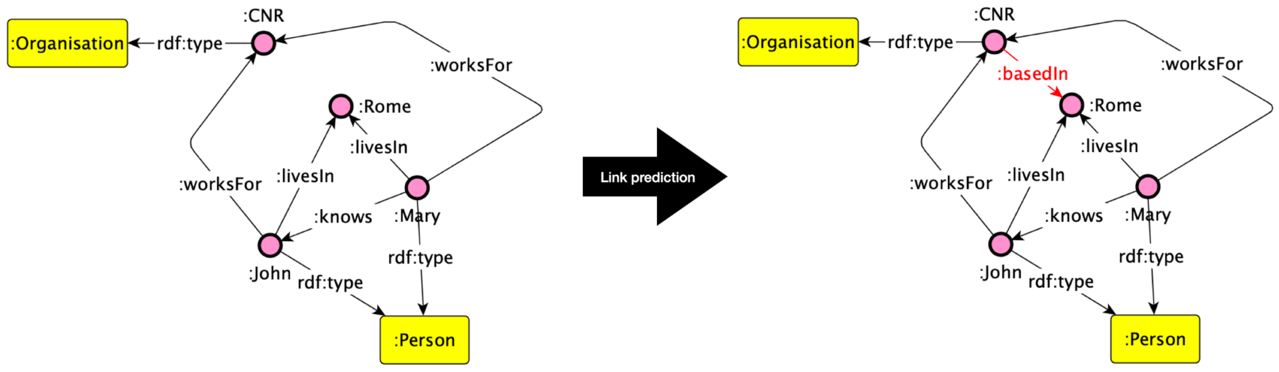


Figure 16: Example of link prediction for enabling intra-linking in RDF graphs.

To solve the link-prediction problem, various techniques have been proposed in literature and there are three main paradigms [20,21], i.e.:

- deterministic or close to deterministic-based predictions which are typical of In inductive Logic Programming (ILP). ILP-based approaches permit the integration of ontological background knowledge. In ILP, the main object of research is not making the predictions per se, but techniques for automatically acquiring components of the theory that enables them, in a logical language;
- Statistical Relational Learning (SRL) approaches that make probabilistic statements formulated either as a conditional probability (as in Probabilistic Relational Models) or as a potential function (as in Markov Logic Programming).
- solutions based on latent representations. Popular examples are RESCAL, TransE, HolE and ComplEx. More recent developments are Graph Convolutional approaches.

Similarly, inter-linking has a number of solutions at the state of the art that enable the generation of alignment axioms (e.g. owl:sameAs, skos:closeMatch, skos:relatedMatch, etc.) between two or more linked open datasetsl. It is possible to reuse the same techniques identified with intra-linking for realising inter-linking. However, a number of frameworks already exist at the state of the art, especially in the domain of knowledge integration. Examples are LIMES [22] and Silk [23], which are two open-source frameworks for RDF link detection. LIMES is a time-efficient approach for link discovery in metric spaces. LIMES utilises the mathematical characteristics of metric spaces during the mapping process to filter out a large number of those instance pairs that do not suffice the mapping conditions. Silk is a link discovery framework that features a declarative language for specifying which types of RDF links should be discovered between data sources as well as which conditions entities must fulfil in order to be interlinked. Link conditions may be based on various similarity metrics and can take the graph around entities into account, which is addressed using a path-based selector language. Silk accesses data sources over the SPARQL protocol and can thus be used without having to replicate datasets locally.

Examples of external sources that WHOW aims at linking with are Copernicus[46] and other pertinent datasets from the European Data Portal[47] (EDP). The linking with Copernicus and data available in EDP can be enabled by their SPARQL endpoints or, in case of Copernicus, by integrating its API into LIMES or Silk.

The action *"7. Enrichment"* in Figure 10 implements the *DataEnrichmentManager* component of the WHOW architecture described in the deliverable 4.1 [10]. Accordingly, such an action addresses the requirement *FR-01* (cf. [10]), which is focused on supporting the definition of automated data cleaning, processing, transformation and publication pipelines involving datasets originating from the data providers' internal/existing systems and from external data sources. Similarly, the actions *"8a. Prediction of internal links"* (i.e. intra-linking) and *"8b. Prediction of links with external RDF graphs"* (i.e. inter-linking) implements the DataLinker component of the architecture. Hence, they fulfil the specific requirement *FR-17*, which targets the support of data interoperability with a special focus on *"cross-site datasets interlinking and linking to external datasets/knowledge graphs"* [10].

3.5 RDF data publication

Once the data are available as valid and rich RDF, then it is possible to publish those data into the Linked Open Data cloud, i.e. action *"9. Publication as Linked Open Data"*. This action is divided into 4 sub-actions that take care about the publication under different lenses, that is:

- *10a. Storage in triplestore*. The RDF data are loaded on a triplestore persistently by also relying on named graphs for associating different RDF to as many unique graphs identified by IRIs. A triplestore is a type of graph database that stores semantic facts modelled as RDF. We want to rely on state of the art triplestores for implementing this action. Examples are OpenLink Virtuoso[48], Stardog[49], GraphDB[50], etc. This addresses the requirement FR-0401[51], FR-05[52] and implements the component of the WHOW architecture named *Triplestore*. We remand the interested reader to the deliverable on the architecture [10] for more details.
- *10b. SPARQL endpoint setting-up*. The RDF data must be accessible via SPARQL queries both to software agents and humans. Hence, a SPARQL endpoint needs to be set up for supporting such a query formalism. Typically, state of the art triplestores also come along with SPARQL endpoints that enable query mechanisms through the HTTP(s) protocol. This action implements the

[46] https://www.copernicus.eu/en
[47] https://data.europa.eu/en
[48] https://virtuoso.openlinksw.com/
[49] https://www.stardog.com/
[50] https://graphdb.ontotext.com/
[51] FR-0401: "Data shall be stored in the TripleStore.".
[52] FR-05: *"The TripleStore shall organise data in named graphs."*

SPARQLEndpoint component of the WHOW architecture [10] and addresses the requirements FR-37[53] (and all its sub requirements), FR-38[54], FR-39[55], FR-40[56], FR-41[57], and FR-42[58].

- *10c. HTML-based view.* This action implements the *DataBrowser* component of the WHOW architecture [10], which is a Web-based application that allows users to visualise and navigate Linked Open Data by using their preferred browser and provided visual metaphors and patterns. Hence, this action addresses specifically the requirements: FR-18[59], FR-19[60], FR-20[61], FR-21[62], FR-23[63], FR-24[64], FR-35[65], and FR-36[66]. A common software solution to data browsing in the Linked Open Data community is LodView[67]. Hence, we want to rely on this solution for implementing the data browser in WHOW.
- *10d. API publication.* This action specifically targets the functional requirement FR-43 that enables WHOW to expose a set of REST APIs, compliant with the ontology network, for data access and consumption, documented in compliance with the OpenAPI Specification (OAS). The action contributes to the implementation of the *APIManager* component of the WHOW architecture and its related requirements [10].

[53] FR-37: *"The platform shall provide an access service that is able to support SPARQL protocol requests and possibly SPARQL protocol extensions such as GeoSPARQL."*.
[54] FR-38: *"The platform shall provide the user with a UI for querying the Knowledge Graph."*.
[55] FR-39: *"The platform shall support the execution of federated queries over the knowledge graph and across multiple sites (cross-site queries and queries involving external endpoints)."*.
[56] FR-40: *"The platform shall provide access to the open Knowledge Graph available at each site via one or more SPARQL endpoints."*.
[57] FR-41: *"The human agent shall be able to perform custom queries on the Knowledge Graph via a human interaction service."*.
[58] FR-42: *"The human agent shall be able to select a predefined SPARQL query from a set of example queries provided through a UI."*.
[59] FR-18: *"The platform shall visualise information based on a provided lens that organises and aggregates data according to a given view."*.
[60] FR-19: *"The platform shall provide one with a number of filters based on the knowledge modelled by the ontology network and its modules, e.g. for enabling faceted browsing capabilities.."*.
[61] FR-20: *"The user shall be able to explore the Knowledge Graph incrementally by means of visual metaphors."*.
[62] FR-21: *"The user shall be able to retrieve information about a resource by giving its IRI."*.
[63] FR-23: *"The user shall be able to visualise type-specific properties of an instance."*.
[64] FR-24: *"The user shall be able to filter out and get details of the data according to specific users' requests."*.
[65] FR-35: *"The platform shall support content negotiation."*.
[66] FR-36: *"All the resources shall be dereferenced by https."*.
[67] https://lodview.it/

4 How to feed and automate the WHOW linked open data production process: the cases of ISPRA ad ARIA Spa

The two WHOW data providers directly involved in WHOW as project partners (i.e. ISPRA and ARIA Spa) show different existing open data architectures. While ISPRA already embraces the Linked Open Data paradigm and thus deploys all the elem ents that are necessary to produce LOD, Aria in contrast manages open data processes that feed as ultimate goal an open data catalogue. There is no current systematic management of linked open data.

In the light of these scenarios, we propose two ways to plug the WHOW linked open data production process in existing open data systems.

4.1 WHOW LOD process at Aria SpA

A totally different scenario is to be faced in Aria. As previously reminded, currently Aria does not deploy any linked open data platform. In this case, it is necessary to plug in the entire WHOW linked open data production process we designed. Specifically, it has been decided to plug in the various steps forming the WHOW LOD process at the end Aria's open data process, after the publication of the datasets in the open data catalogue of Lombardy Region. This strategy guarantees that:

- no other parallel processes for open data is started with respect to the main ones already implemented and deployed;
- input files are available to be transformed in the WHOW knowledge graph;

- the entire process is sustainable over time: the linked open data production process is fully integrated in the data governance of Lombardy Region; an additional representation of the data can then be published in the related open data catalogue.

However, in order to first locate the datasets in the open data catalogue and create the RDF datasets with as minimal manual interventions as possible in the process, we propose the following strategy for the design of the WHOW LOD production at Aria.

The strategy has been already adopted in a European Commission pilot project conducted in a different application domain, i.e., the procurement domain. In this pilot, the objectives were similar to ours: create a sustainable knowledge graph production process where any manual interventions are minimised.

The strategy foresees to use a novel declarative approach where an **OWL meta level ontology named transformation ontology** is used to manage declarative descriptions of the steps of a possible workflow for LOD production. These descriptions are leveraged **by a workflow engine** that orchestrates the tasks necessary to transform input datasets into the desired RDF representation. In essence, the OWL meta level ontology governs the WHOW LOD process starting from datasets available in open data catalogues, locating the necessary transformation resources (e.g. RML scripts) and producing output RDF representations of those datasets then documented in the open data catalogues as additional datasets' distributions.

Figure 17 illustrates the high-level view of the process that can be deployed at Aria open data infrastructures.

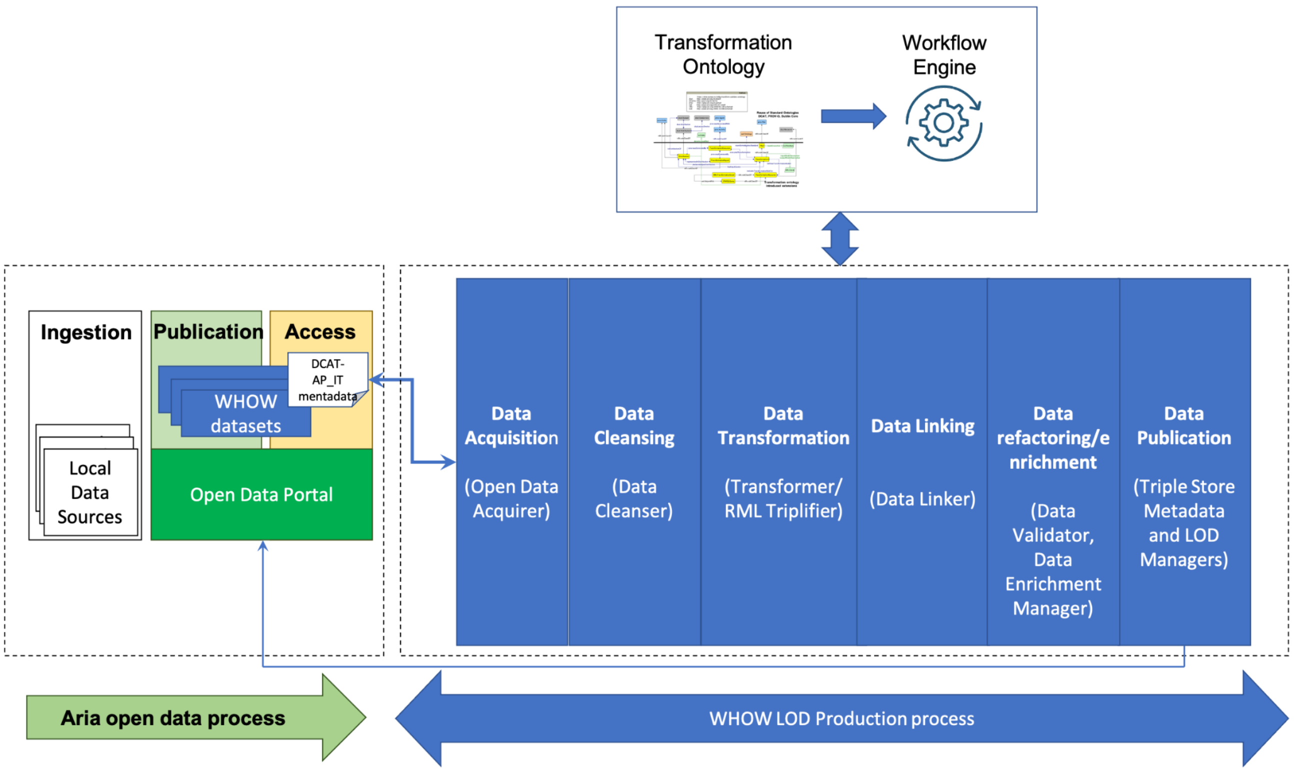

Transformation Ontology
Workflow Engine
Ingestion
Publication
Access
DCAT-AP_IT mentadata
WHOW datasets
Local Data Sources
Open Data Portal
Data Acquisition (Open Data Acquirer)
Data Cleansing (Data Cleanser)
Data Transformation (Transformer/ RML Triplifier)
Data Linking (Data Linker)
Data refactoring/enrichment (Data Validator, Data Enrichment Manager)
Data Publication (Triple Store Metadata and LOD Managers)
Aria open data process
WHOW LOD Production process

Figure 17: Integrating WHOW LOD Process at ARIA open data infrastructures.

Transformation ontology

The OWL ontology that controls the LOD production and validation process is illustrated in Figure 18. The ontology is grounded on two foundational ontologies for metadata description; namely, DCAT-AP - European Application Profile for Data Catalogue Vocabulary, which extends the DCAT Web recommendation[68] in order to describe datasets available in data catalogues, and PROV-O - Provenance Ontology[69], another Web recommendation which allows one to represent all provenance information related to activities and entities. The transformation and validation ontology imports PROV-O and extends it with a minimum set of classes and properties (the bottom level in Figure 18) that represent the specific transformation activities and resources to be done and used in the semantic knowledge graph construction process. The ontology is rather simple, with elements that can be clearly understood in contexts such as the public sector where also the use of DCAT-AP is gaining momentum. Aria in fact adopts the Italian extension, DCAT-AP_IT, of the European profile that can be used to locate the input datasets in the open data catalogue that must be transformed in RDF according to the WHOW ontology network.

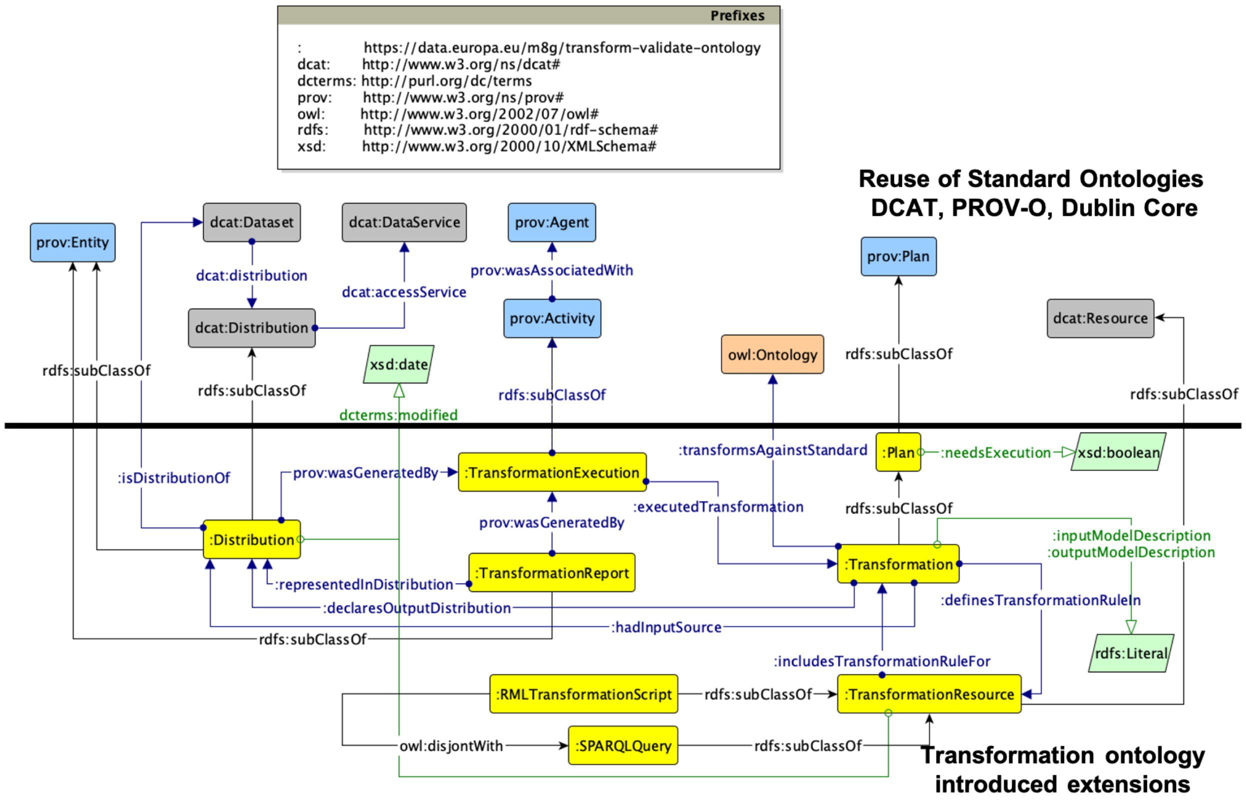


Figure 18: Transformation ontology.

[68] https://www.w3.org/TR/vocab-dcat-2/
[69] https://www.w3.org/TR/prov-o/

Ontology description. A transformation (the class :`Transformation`) is a specific type of PROV-O plan (thus represented as subclass of `prov:Plan)`, and it is defined as a planned set of operations to be executed by one or more agents. Since the triplier architectural component transforms a given input dataset distribution into an output dataset distribution, we identify a dataset distribution by extending the same concept as the one defined in DCAT so as to link it to the core elements of the transformation ontology. For instance, we add an inverse property from our `:Distribution` concept to the `dcat:Dataset` class and an OWL restriction that represents the connection of the distribution of a dataset to the execution of a transformation plan. This extension is represented by the class `:Distribution` (bottom part of Figure 18); it inherits all the properties of the main `dcat:Distribution` (e.g., `dct:modified`, `dcat:accessURL`, etc.), including the relationship with the class `dcat:DataService`. A transformation plan defines transformation rules within specific types of transformation resources (the class `:TransformationResource` intended as a subclass of `dcat:Resource`). A transformation resource can be the RML mapping script (the class `:RMLMappingScript`) we code in order to provide the necessary rules for an RML engine that indicate how to map the input data file in an RDF file represented using the WHOW ontology network (the `owl:Ontology` class in Figure 18). A `:TransformationExecution` activity (a subclass of `prov:Activity`), executed by some Agent (`prov:Agent`), is defined. It generates a dataset distribution (`:Distribution`), executes (the `:executeTransformation` property) a transformation plan and produces a report (the class `:TransformationReport`). This latter is a `prov:Entity` representing any return message that gives information on the success or otherwise of the transformation operation.

Workflow Management System

In order to execute all the different phases of the WHOW LOD process (as shown in Figure 10) at scheduled times and based on activities and resources identified by the instances of the transformation OWL ontology we decided to make use of a workflow management system. The use of such a system guarantees a certain degree of automation when running the overall process. We can identify two main phases:

- Workflow definition, where the instances of the ontologies are used to define tasks and resources in input and produced as output;
- Workflow execution, where a scheduler executes the tasks

Workflow definition. The workflow engine is considered ephemeral, as all state is persisted in the (extended) catalogue. This guarantees a clear separation between the business processes whose output is recorded in the catalogue and the operational side, in the form of code executed by the engine. In this phase of the management of the workflow system, the instances of the OWL transformation ontology are used as input to generate a workflow. In particular, each instance of the `:Transformation` class of the ontology is turned into a workflow object. In its most basic setup, each workflow contains a single Transformation task, which executes RML scripts. The same approach can be used for other activities and tasks.

Workflow execution. When the transformation task is executed, the `:TransformationExecution` class is instantiated. There are then three steps that are performed:

- *Extract*: The file(s) referenced by the `dcat:accessURL` property of the input distribution, (identified using the `:hadInputSource` property of the ontology defined for the `:Transformation` class instance), gets downloaded to the worker nodes of the workflow management system used. The file(s) represent(s) the original dataset(s) usually available in CSV, JSON formats, to be transformed in RDF (see the arrow between the DCAT-AP_IT metadata and the Data Acquisition/Extraction step of Figure 17). Also, the `:RMLMappingScript` instance (since in our case RML-based transformation is carried out), also referenced from the `:Transformation` class instance, is downloaded to the worker nodes of the workflow management system used.

- *Transform:* after properly configuring the files extracted in the previous step, this phase of the execution of the workflow is responsible for (i) executing the transformations through the Triplifier; (ii) launching data linker in order to enable links between the WHOW datasets and others available in the Web of Data; (iii) running the Data validator and enrichment managers in order to possibly refactoring the produced RDF datasets; (iv) storing the results in the triple store through the LOD manager,. In addition, the results of all these steps of the workflow are then saved in files of the workflow nodes.
- *Load*: What has been produced in the previous step is written back to the `dcat:accessURL` of the output distribution of the transformation OWL ontology and an instance of the dcat:DataService class is created in order to document the availability of the RDF based knowledge graph in the SPARQL endpoint for querying purposes.

4.2 WHOW LOD process at ISPRA

The workflow illustrated in Figure 6 and the related deployment architecture in Figure 7 also apply to the WHOW LOD process. The platform implemented in ISPRA is natively aimed at producing Linked Open Data, which guarantees that any other dataset to be transformed into RDF can undergo the same processing steps described above. In this sense, the WHOW perfectly fits the case of ISPRA. Hence, datasets contributing to use cases in which ISPRA plays a role will be extracted through the Data Extraction component of the Data Extraction Nodes and presented as CSV files to the triplifier. The latter transforms the input datasets into RDF datasets that can be loaded into the SPARQL endpoint and documented through the metadata DCAT-AP_IT and GeoDCAT-AP_IT. Since ISPRA's triplifier uses the RML mapping language, no specific modifications to this component are necessary: the RML scripts produced in the context of WHOW are used by the triplifier engine used at ISPRA to produce the RDF datasets.

More pragmatically, ISPRA framework already implements the *"1. Data preparation and cleansing"* and *"2. Triplification"* activities shown in the workflow in Figure 10. The former is extensively described into the WHOW deliverable 3.1 [9], the latter is implemented on top of RML by using pyRML[70]. pyRML is a Python based engine for processing RML files. We remind the reader that the RDF Mapping Language (RML) is a mapping language defined to express customised mapping rules from heterogeneous data structures and serialisations to the RDF data model. RML is defined as a superset of the W3C-standardised mapping language R2RML, aiming to extend its applicability and broaden its scope, adding support for data in other structured formats. In this sense, RML allows the ISPRA framework to address the triplification designed in workflow presented in the current document (cf. Figure 10) by applying a mapping-based solution that allows to specify custom transformation rules for each different component of a triple, i.e. subject, predicate, and object. Thus this addresses the actions *"3a. Subject generation"*, *"3b. Predicate generation"*, and *"3c. Object generation"* presented in Figure 10 that are preparatory to the action *"4. Triple generation"*.

The RDF data validation, i.e. action *"5. Validation"* in Figure 10, is currently addressed by ISPRA by using a predefined set of SPARQL queries for checking the result sets. This can be extended by adding a DL reasoner

[70] https://github.com/anuzzolese/pyrml

for consistency check and error provocation, and a SHACL[71] engine for assessing the quality of data in terms of shape graphs, also known as "*data graphs*".

The RDF data enrichment, i.e. action *"7. Enrichment"* in Figure 10 is implemented in ISPRA by means of LIMES[72] (Link Discovery Framework for Metric Spaces) [24], which is a framework for discovering links between entities contained in Linked Data sources.

Finally, RDF data publication addresses the workflow presented in Figure 10 in the following way: the action *"10a. Storage in triplestore"* and *"10b. SPARQL endpoint setting-up"* relies on OpenLink Virtuoso[73] for their implementation, while *"10c. HTML-based view"* relies on LODView[74], which is a Java web application based on Spring and Jena. LodView, in conjunction with a SPARQL endpoint, allows a user to access RDF data as HTML based representations of RDF resources. The *"10d. API publication"* is currently missing and needs to be implemented.

[71] https://www.w3.org/TR/shacl/
[72] https://github.com/dice-group/LIMES
[73] https://virtuoso.openlinksw.com/
[74] https://github.com/LodLive/LodView

5 Conclusions

This deliverable discussed the current state of the art of the open data systems implemented and distributed by the two data providers of the WHOW project. In particular, different scenarios were considered: while ISPRA has already adopted the linked open data paradigm for the publication of some of its open data, Aria has not yet adopted it but provides a well-established pipeline of level 3-star open data, according to the open data star scale proposed by Tim Berners Lee[75]. The pipeline ends with the publication of open datasets in a rich open data catalogue, where all datasets are accompanied by metadata compliant with DCAT-AP(_IT) version 1.1.

The state of the art of the currently available systems was instrumental in understanding how to incorporate the LOD production process of the WHOW project into these systems. In this regard, the deliverable introduces the design of the WHOW process by proposing high-level views and more detailed UML activity diagrams in which the different steps of the process are described. Furthermore, for each data provider, the deliverable discusses and proposes solutions to integrate the WHOW LOD pipeline. In general, a declarative solution is proposed in the deliverable, which has also been adopted in European pilot projects for other application domains. The solution makes use of an OWL ontology called transformation ontology, developed by ISTC-CNR researchers, and of a workflow management system that uses instances of the ontology to govern the different tasks to be performed to extract, transform and load the various open datasets..

It is worth highlighting that this deliverable is the basis for future development work on the WHOW project, some of which has already begun with an initial set of open datasets identified in the three WHOW use cases. The design of the process, in an agile spirit, may be subject to future evolutions according to the different requirements and needs that may emerge in the implementation phases.

[75] https://5stardata.info/en/

References

1. Bizer, Christian, Tom Heath, and Tim Berners-Lee. "Linked data: The story so far." Semantic services, interoperability and web applications: emerging concepts. IGI global, 2011. 205-227.
2. Daga, Enrico; Asprino, Luigi; Mulholland, Paul and Gangemi, Aldo (2021). Facade-X: An Opinionated Approach to SPARQL Anything. In: Alam, Mehwish; Groth, Paul; de Boer, Victor; Pellegrini, Tassilo and Pandit, Harshvardhan J. eds. Volume 53: Further with Knowledge Graphs, Volume 53. IOS Press, pp. 58–73.
3. Delva, Thomas, et al. "Integrating Nested Data into Knowledge Graphs with RML Fields." *Proceedings of the 2nd International Workshop on Knowledge Graph Construction (KGCW 2021)*, vol. 2873, 2021.
4. De Meester, Ben, et al. "Mapping Languages: Analysis of Comparative Characteristics." *Proceedings of First Knowledge Graph Building Workshop*, 2019.
5. García-González, Herminio, "A ShExML perspective on mapping challenges: already solved ones, language modifications and future required actions", *ESWC 2021 Workshop KGCW Submission*, 2021.
6. Iglesias-Molina, Ana, Andrea Cimmino and Oscar Corcho. "Devising Mapping interoperability with Mapping Translation". *Proceedings of the Third International Workshop on Knowledge Graph Construction*, 2022.
7. Lippolis, Anna Sofia, Giorgia Lodi and Andrea Giovanni Nuzzolese. "Design of the technical services for knowledge graph management" (technical report), 2021.
8. Wilkinson, Mark D., et al. "The FAIR Guiding Principles for scientific data management and stewardship." Scientific data 3.1 (2016): 1-9.
9. Anna Sofia Lippolis, Giorgia Lodi, Andrea Giovanni Nuzzolese, Gianluca Carletti, Elio Giulianelli, Marco Picone, Giulio Settanta. Data pre-processing is finalised. Deliverable 3.1 of the WHOW Project (EU CEF Telecom INEA/CEF/ICT/A2019/2063229) Prot. CNR-ISTC 1220/2022. 2022 https://github.com/whow-project/deliverables/blob/main/WHOW_Del%203.1_DataPreprocessing.pdf
10. Anna Sofia Lippolis, Giorgia Lodi, Andrea Giovanni Nuzzolese. Design of the technical services for knowledge graph management. Deliverable 4.1 of the WHOW Project (EU CEF Telecom INEA/CEF/ICT/A2019/2063229) Prot. CNR-ISTC 0000203/2022. 2021 https://github.com/whow-project/deliverables/blob/main/ArchitectureDeliverable_final.pdf
11. Dimou, Anastasia, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "RML: a generic language for integrated RDF mappings of heterogeneous data." In Ldow. 2014.

12. Lefrançois, Maxime, Antoine Zimmermann, and Noorani Bakerally. "A SPARQL extension for generating RDF from heterogeneous formats." In European Semantic Web Conference, pp. 35-50. Springer, Cham, 2017.
13. García-González, Herminio, Iovka Boneva, Sławek Staworko, José Emilio Labra-Gayo, and Juan Manuel Cueva Lovelle. "ShExML: improving the usability of heterogeneous data mapping languages for first-time users." PeerJ Computer Science 6 (2020): e318.
14. Gangemi, Aldo, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. "Modelling ontology evaluation and validation." In European Semantic Web Conference, pp. 140-154. Springer, Berlin, Heidelberg, 2006.
15. Glimm, Birte, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. "HermiT: an OWL 2 reasoner." Journal of Automated Reasoning 53, no. 3 (2014): 245-269.
16. Carriero, Valentina Anita, Fabio Mariani, Andrea Giovanni Nuzzolese, Valentina Pasqual, and Valentina Presutti. "Agile Knowledge Graph Testing with TESTaLOD." In ISWC Satellites, pp. 221-224. 2019.
17. Blomqvist, Eva, Valentina Presutti, Enrico Daga, and Aldo Gangemi. "Experimenting with eXtreme design." In International Conference on Knowledge Engineering and Knowledge Management, pp. 120-134. Springer, Berlin, Heidelberg, 2010.
18. Blomqvist, Eva, Karl Hammar, and Valentina Presutti. "Engineering Ontologies with Patterns-The eXtreme Design Methodology." Ontology Engineering with Ontology Design Patterns 25 (2016): 23-50.
19. Hogan, Aidan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane et al. "Knowledge graphs." Synthesis Lectures on Data, Semantics, and Knowledge 12, no. 2 (2021): 1-257.
20. Nickel, Maximilian, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A review of relational machine learning for knowledge graphs." Proceedings of the IEEE 104, no. 1 (2015): 11-33.
21. Rettinger, Achim, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. "Mining the semantic web." Data Mining and Knowledge Discovery 24, no. 3 (2012): 613-662.
22. Ngomo, Axel-Cyrille Ngonga, and Sören Auer. "LIMES—a time-efficient approach for large-scale link discovery on the web of data." In Twenty-Second International Joint Conference on Artificial Intelligence. 2011.
23. Volz, Julius, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. "Silk-a link discovery framework for the web of data." In LDOW. 2009.
24. Ngomo, Axel-Cyrille Ngonga, and Sören Auer. "LIMES—a time-efficient approach for large-scale link discovery on the web of data." In Twenty-Second International Joint Conference on Artificial Intelligence. 2011.