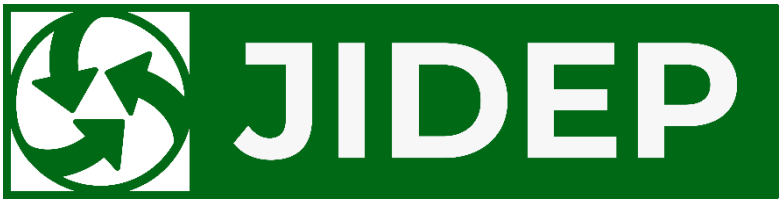
PROJECT DELIVERABLE REPORT

Grant Agreement Number: 101058732



Joint Industrial Data Exchange Platform

Type: Other

D3.2 Report on Tools Developed for Schema and Dataset Alignment and Knowledge GraphDevelopment

Issuing partner	UNITN
Participating partners	UCAM, TVS
Document name and revision	D3.2 Report on Tools developed for schema and dataset alignment and KG development
Author(s)	Simone Bocca Rasel Ahmed (TVS)
Deliverable due date	2024-June-01
Actual submission date	

Project Coordinator	Vorarlberg University of Applied Sciences
Tel	+43 (0) 5572 792 7128
E-mail	florian.maurer@fhv.at
Project website address	www.jidep.eu

Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	
SEN	Sensitive, limited under the conditions of the Grant Agreement	✓

Content

1. Introduction

1.1 Executive Summary

This is the first version of the deliverable describing how the Knowledge Graph (KG) generation process, defined by the iTelos methodology, is supported by a dedicated framework, composed of different tools. In the context of the JIDEP European project, the iTelos framework aims to provide a concrete implementation of the iTelos methodology for the generation of reusable KGs. The framework, here described, supports the implementation of the methodology executed by a human user. Nevertheless, some of the tools, composing the whole framework, are designed to be integrated into the JIDEP Distributed Ledger Technology (DLT) platform (see D3.4) as semi-automatic tools. The different tools aim at producing different intermediate outputs, along the iTelos phases. The key idea, introduced by the iTelos methodology, is to compose the different tools, by considering as input of each tool, the output of the tool executed previously, following the methodology. Such an approach, in the iTelos implementation, aims at reducing the effort in the execution of the different activities required to build a KG. Moreover, the objective is to provide a complete set of tools for KG generation, starting from the formalisation of the requirements, until the generation of a unique object (file or set of files) representing the final KG. The current deliverable reports the description of each tool adopted along the iTelos methodology, as well as how to use them to achieve the desired results.

The current section of the deliverable reports the executive summary of the document together with the description of the intended audience. Section 2 describes the implementation of the iTelos process, by defining the requirements to be satisfied at each phase of the methodology. Section 3 describes which are the different tools included in the iTelos framework, adopted to satisfy the requirements described in the previous section. In this section the deliverable reports how the tools are structured, which is the objective for which they have to be used, together with their inputs, outputs and the usage instructions. Sections 4 concludes the deliverable with the conclusions, and section 5 provides a summary of the details that have been updated from the previous version of the deliverable.

1.2 Intended Audience

The intended audience for this report is composed of the following subjects:

- JIDEP project technical roles: subjects involved in the JIDEP project, having technical competencies, and mainly working over the implantation of the DLT platform.
- Data scientists: technical subjects involved in the iTelos methodology process, by acting over the data management activities (data collection, cleaning and formatting).

- Knowledge engineers: technical subjects involved in the iTelos methodology process, by acting over the knowledge management activities (reference ontology collection, schema modelling, schema alignment).

2. The iTelos Process

In this section, it is reported a description of the whole process supporting the iTelos methodology described in D3.1 (final version). The process is supported by a dedicated framework, composed of different tools which aim at supporting the different phases of the methodology for KGs generation. For each iTelos phase, there are requirements to be satisfied. Such requirements need to be covered by the tools adopted to implement the KG generation process. The sub-section 2.1 aims to describe which are the requirements to be satisfied for each iTelos phase, to clarify the usage of the tools in the relative phases.

2.1 iTelos Process Requirements

The requirement set for the entire process implementing the iTelos methodology is distributed along the different iTelos phases. Each phase has its specific objective in the KG generation process. For this reason, within each phase, a specific set of sub-requirements needs to be satisfied in order to provide the proper input to the next phase, thus proceeding with the whole process. Here below is the list of all the main requirements to be satisfied, together with the reference iTelos phase in which they have to be achieved. In the next section, the report will describe how these requirements are satisfied by using specific tools.

- **Formalisation of the Initial Purpose**: the first key requirement to be satisfied is the formalisation of the initial purpose. This requirement is included in the first phase of the methodology. Nevertheless, in specific cases, where a proper level of formalisation for the initial purpose is difficult to achieve, this requirement can be satisfied by a dedicated phase of the methodology (see the final version of D3.1 for a complete description of the iTelos methodology). The key idea is to transform the purpose, initially stated by the users as a natural language sentence, into a set of formal requirements and information models, to be adopted in the next iTelos phases for the generation of the final KG. Such an objective is achieved by the iTelos methodology by producing a set of Competency Questions (CQs), and an ER model representing the information to be considered in the final KG.

- **Data Collection**: in the second phase of the methodology (always considering the iTelos methodology as it is described by the final version of the deliverable D3.1), the main requirement to be satisfied is to collect a set of resources to be then exploited in the next phases in order to generate the final KG. Nevertheless, two different sub-requirements are included in the main one. (i) The first one is related to the type of resources collected. In fact, both data and knowledge resources need to be collected in this phase. In other words, in the set of collected resources are included both datasets and datasets schemas (ontologies). (ii) The second sub-requirement to be

considered is the identification of the sources (websites, web services, databases and many others) from which the resources have to be collected. This activity is not trivial due to the heterogeneity present in the set of available information sources. Such heterogeneity is compounded by the different ways adopted by the source to represent and describe the resources they distribute. The resources can be described by the sources using different information (metadata), thus can be difficult to recognise if a resource is suitable for the purpose to be achieved. Moreover, the resources can represent the information distributed by adopting different formats (CSV, TSV, Excel, PDF, and many others) thus introducing more effort in converting the data to be collected in the same format, suitable to be processed along the next phases of the iTelos methodology.

- **Domain Languages Alignment**: during the third phase of the methodology, the languages (natural languages and domain languages) adopted to represent the information in the final KG, play a crucial role. The key idea is to adopt standard and “well-known” domain languages, where the concepts expressed are precisely defined in the domain of interest, to increase the interoperability of the final KG. To this end, it is crucial to know if the concepts (represented by domain language terms) to be adopted in the KG, are already defined in reference domain-specific vocabularies (or ontologies) or not. In case such concepts are already expressed by existing standard vocabularies, they will be “annotated” as existing concepts referring to the relative standard meaning. If instead, the interested concepts are not available in any of the standard vocabularies considered, they will be “annotated” as new concepts. The KG produced by using such annotated concepts will be more interoperable and reusable, due to the adoption (whenever possible) of standard concepts related to the specific purpose. To satisfy such a requirement it is necessary to collect and disambiguate all the concepts used to build the KG project, providing for each concept a description, an identifier as well as the corresponding reference knowledge source (vocabulary or ontology), if it exists.

- **Schema Alignment**: in the fourth phase of the iTeos methodology the main requirement considered is again referred to the interoperability and reusability of the final KG. Nevertheless, this case is focused on the final KG schema, instead of the language adopted to represent the information included. The key idea is to build the purpose-specific ontology (see D2.3 for more detail about the ontology generated) of the final KG, by reusing as much as possible (portions of) standard domain-specific reference ontologies. To satisfy such a requirement, it is necessary to use a tool that is able to support the analysis of existing ontologies to understand if the conceptualisations they provide, for the relative domain of interest, can be reused to build the purpose-specific ontology that defines the structure of the final KG. Moreover, there is the need to support the modelling of such purpose-specific ontology.

- **Data Mapping**: in the last iTelos phase the main requirement is to map the datasets collected with the purpose-specific ontology defining the structure of the final KG. To this end, another sub-requirement needs to be considered. The datasets collected have to be properly updated in order to align the data types with those specified in the purpose-specific ontology. Considering the two requirements described above, the last phase of the methodology needs to be supported by a tool able to perform the data types modification over the datasets, as well as to merge such datasets with the purpose-specific ontology, thus building, as final output, the KG.

The last crucial aspect to be considered, defining the requirements for the framework supporting the iTelos methodology, is the need to compose the above-described requirements into a single process, in such a way that the satisfaction of one of them depends on the satisfaction of the requirements that come before. This is a crucial aspect considering that the execution of the iTelos process can proceed backwards in case of dissatisfaction with the evaluation activities, which decide if the intermediate output of each phase is good enough to be used in input for the subsequent phase. A set of tools suitable to support such a process has to be able to work connected into a framework where the output of a specific tool has to be properly executed, with a limited effort, as the input of the subsequent one.

3. The iTelos Framework

In this section, the iTelos framework is detailed, by describing the specific tools applied in each phase of the methodology. More in detail the description of the tools is based on the requirements to be satisfied at each phase (see Section 2.1). Due to that, in some iTelos phases, the usage of more than one tool is required to satisfy the relative requirement set.

The goal of the iTelos framework is to compose together the execution of different tools, performing different activities, but having the same common objective, to build a purpose-specific KG. The tools adopted to implement the iTelos methodology have been chosen, and need to be used, to produce the relative intermediate output for each phase in which they have to be adopted, respectively. The key idea is that the output of each tool has to be the input of the tool to be executed subsequently. This kind of connection between the different tools reduces the effort of the overall process of KG generation, by reducing the management of the resources handled at each phase, to be provided as input to the next phases. Here below, each iTelos tool is described, by providing, for each of them, the tool's objective, its expected input and output, as well as the details about its composition and usage instructions.

Moreover, in the last sub-section, it is reported the description of the KG-builder tool. Such a tool is an updated version of the data mapping tool adopted in the last iTelos phase. The KG-builder tool aims at generating KGs through a semi-automatic process.

3.1 Purpose Formalisation Tools

In the initial phase of iTelos, as detailed in section 2.1, the main requirement to be satisfied is to formalise the initial user purpose. The input of this phase is the

purpose as provided by the user, thus usually represented as a natural language statement. The first tool adopted to formalise such an input is a document writer tool used to create the *Purpose Formalisation document* where the four main elements implicitly included in the input purpose, need to be reported:

1. *The Purpose Domain*: the description of the domain of interest. More in detail, the description of the domain defines in general the environment in which the user purpose is defined, thus providing important details about the information to be included (or not) in the final KG.

2. *A Set of Scenarios*: in the same document, a set of different scenarios is defined within the boundaries of the previously defined purpose domain. The scenarios define different details about the possible background environments implicitly included in the purpose domain (and thus in the initial purpose). More scenarios are defined in the Purpose Formalisation document, the more detailed will be the information to be included in the final KG.

3. *A List of Personas*: the description of different personas acting within the different scenarios. The personas defined are the actors considered by the initial purpose. They can be users of the final KG and/or people associated with the information that the final KG needs to contain. The More heterogeneous the description of the personas considered by the purpose will be, the more information will be available to understand how to properly model the final KG.

4. *A List of Competency Questions (CQs)*: having the purpose domain, the scenarios and the personas, the last element to be added in the Purpose Formalisation document, is a list of Competency Questions. It is a list of statements (in question form) describing how one or more personas interact within a scenario (with other personas or with the environment). A CQ represents a single specific use case, to be satisfied by the final KG, included in the initial purpose. For this reason, at the end of the methodology, the CQs will be transformed into queries, to evaluate the final KG.

In the Purpose Formalisation document, the key idea is to extract all the possible details from the informal definition of the initial purpose. To this end, the four main elements, described in the document, aim at specifying as much as possible such information, by defining in the end a list of functional requirements to be satisfied, in the form of CQs. Any text editor can be used to achieve such intermediate output, nevertheless, the four main elements have to be present in the final version of the document. For this reason, a document template can be adopted to support the definition of the required steps.

In the Purpose Formalisation phase (see final version of D3.1 as methodology reference) the second tool adopted is a spreadsheet editor. The objective to be achieved by using such a tool is the extraction of all the “concepts” to be used for modelling the purpose-specific KG’s structure. More concretely the concepts are terms (words, or labels) extracted by the CQs defined previously in the Purpose Formalisation document. Such terms identify the information elements to be included

in the final KG. The spreadsheet produced, named Purpose Formalisation sheet, reports the IDs of the scenarios and persons, as well as of the CQ from which each concept has been extracted.

At the end of the purpose formalisation process, the last tool adopted aims at modelling the initial draft of the final KG's schema, shaped as the Entity Relation (ER) model. To this end, the tool adopted is yEd [1], a diagram editor offering all the building blocks and symbols to design ER models. To obtain the desired ER model, the Entities, with their properties, and the Relations between the Entities considered, are named in the model, by using the concepts extracted previously and collected in the Purpose Formalisation sheet. A portion of the ER model produced at the end of the first phase of the methodology is depicted in Figure 1.

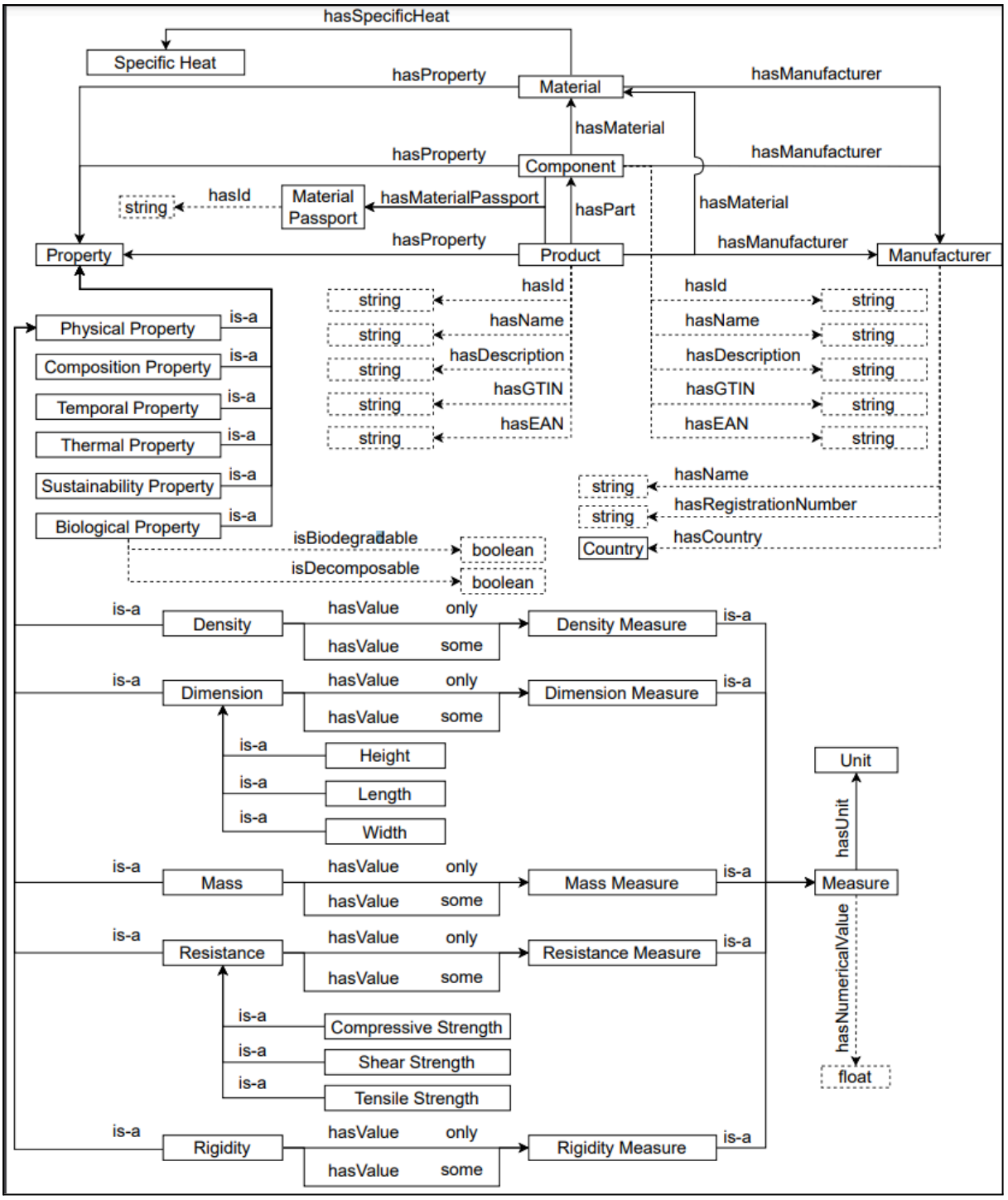


Fig. 1 - Automotive ER model

3.2 Data collection tools

In the second iTelos phase, the objective is to collect the resources required for the creation of the final KG. The data collection activity is often composed of different tasks, depending on the type(s) of data and data sources considered. The heterogeneity at the data source level brings the need to consider different solutions

to concretely extract the required data, from the sources identified. Before the extraction, the need is to identify the sources of data and knowledge, from which the required information can be extracted. Sometimes such sources are provided in input to the iTelos methodology. This is the JIDEP case, where the data to be transformed into reusable KGs, are directly provided by the industrial partners. Nevertheless, that is not always the case. Sometimes the sources have to be found by searching on the web. To support such research of information sources, the iTelos users can exploit the ER model produced in the previous phase, which highlights which are the entities that should be collected to be part of the final KG.

The sources available can be very heterogeneous. Some of them distribute quality data that are described by well-designed metadata, thus improving the findability of such resources. The effort required to extract data from these kinds of sources is usually reduced. Nevertheless, there are data sources, containing useful data, distributed with a low quality (noisy data) and described by using poor metadata. For these second types of sources, the effort to be paid for the data collection is higher. In fact, the collection of data, in this case, includes also the cleaning, filtering and formatting of the data collected. Due to the need to handle the data and its relative source, the tool supporting the data collection offered by iTelos can be divided into two categories:

- **Source Identification and Data Collection.** iTelos offers a data distribution system based on different data catalogues. Such catalogues are web portals where the resources produced by an iTelos execution (KGs, but also ontologies and vocabularies data), are distributed (with the consent of the owner), thus being available for further reuse. Each catalogue distributes data for a specific domain of interest, and all the catalogues are linked and accessible by a main data catalogue. This approach reduces the effort in the source identification activity. Moreover, the catalogues provide different approaches for the data collection (download), depending on the level of privacy and accessibility defined for each dataset published on the catalogues. More information regarding the data catalogues and data search services can be found in the dedicated JIDEP deliverable D3.3.

- **Data Filtering, Cleaning and Formatting**. When building a KG, the data collected comes from different sources. Even considering the above data sources, supporting the iTelos methodology and thus distributing quality data, only a portion of the resources required for the final KG are collected from those sources. Another portion of data (that is not fixed, it depends on the purpose to be satisfied) is collected from low-quality sources, thus providing data that needs to be filtered, cleaned and formatted to obtain the information to be included in the final KG. To this end, iTelos provides a set of well-known data management libraries. The key idea is to facilitate the filtering, cleaning and formatting of sub-activities. Nevertheless, such activities always require experience in programming languages (Python) and data management, for this reason, if the iTelos user doesn’t have such skills, the user needs to be

supported by a data scientist. Here below is a list of the most common Python based libraries adopted by iTelos to filter, clean and format the data collected.

- Dataframe management: Pandas
- Array and matrix data management: NumPy
- Plotting: Matplotlib, graphviz
- REST API: Requests
- Data scraping: Scrapy, BeautifulSoup4
- Dates values management: Dateparser, Arrow
- Geospatial data management: GeoPandas, geopy, geoplot (Cartopy)

The above libraries can be used in different ways, due to the fact that they offer a quite large range of options to handle the data collected. Nevertheless, the iTelos methodology defines clearly which should be the criteria to be considered during the filtering, cleaning and formatting activities. More in detail, the key idea is to handle the data collected by:

- filtering out the portions of information that do not give any support for the satisfaction of the main purpose;
- cleaning the remaining data, by avoiding noisy values, like missing values, typos, and duplicated values;
- formatting the above cleaned data, by unifying the data types and representation formats (i.e., using the same format to represent date values).

The above activities are executed to both the data and knowledge resources. The resources obtained after the above activities need to be properly defined to be used as input for the next iTelos phases. For this reason, the methodology defines clearly the formats in which such resources have to be produced, which are the JSON and RDF-OWL formats, for data and knowledge resources, respectively.

3.3 Domain Languages Alignment Tool

The main requirement for the third phase of the iTelos methodology, As already mentioned in section 2.1, is to collect and disambiguate the concepts used to express the information into the KG produced at the end of the methodology execution. By considering the methodology contexts, a concept can be defined as the base element of a vocabulary (or a terminology) for a specific domain of interest. In other words a concept is a term (or a word) used to describe a specific part of the information into the relative domain of interest (i.e., automotive sector, wind turbine sector, PCB sector). Nevertheless, a concept can assume different meanings (or senses) depending on how, and who, is using it. For example, the concept expressed by the word "wheel" can be used to express information about both a car wheel and a truck wheel, even though the characteristics of the wheels for the two objects are different. For this reason, it is important to define each concept precisely, with the aim of disambiguating the meaning of each term. Any type of data that is built using such disambiguated concepts to express its information becomes more

understandable to the users who need to exploit it, thus making the data even more interoperable and reusable.

The KG construction process defined by iTelos identifies the concepts to be disambiguated into the datasets as well as in the reference ontologies, collected in the second phase (see the section 3.2). More in detail, such concepts are used to define the following elements.

- The ETypes, or ontology classes, used to define the type of real world entities to be represented into the KG produced. Each EType is named with one, and only one, concept.
- The EType's properties are named by using specific concepts.
- The value used to identify each real world entity can be a concept that needs to be disambiguated.
- The values of the entity's attributes can be represented by specific attributes that need to be disambiguated to better understand their meaning.

1	Concpet ID	Word	Knowledge Source	Knoweledge Type	Description
2	JIDEP-0001	Product	http://www.theworldavatar.com/kg/ontomatpassport	Class	An artefact manufactured in an industrial setting to meet the demands of consumers and usually consists of multiple parts.
3	JIDEP-0002	Component	http://www.theworldavatar.com/kg/ontomatpassport	Class	Part of a bigger whole that can be a system or a product of composite nature.
4	JIDEP-0003	Material	http://emmo.info/emmo	Class	A matter individual that stands for a real world object representing an amount of a physical substance (or mixture of substances) in different states of matter or phases.
5	JIDEP-0004	Material passport	http://www.theworldavatar.com/kg/ontomatpassport	Class	A passport or document providing identification, physical, temporal, sustainability and other properties of a product or component.
6	JIDEP-0005	Agent	http://xmlns.com/foaf/0.1	Class	A person, group, software or physical artifact
7	JIDEP-0006	Country	https://dbpedia.org/ontology	Class	A political entity associated with a territory considered a motherland to a nation ruled by a government.
8	JIDEP-0007	Image	http://purl.org/dc/dcmitype	Class	A visual representation of an inanimate or animate object captured through technological means or drawn by humans.
9	JIDEP-0008	Manufacturer	http://www.theworldavatar.com/kg/ontomatpassport	Class	A business, company or entity manufacturing goods for the consumer market.
10	JIDEP-0009	Supplier	http://www.theworldavatar.com/kg/ontomatpassport	Class	A business, company or entity supplying goods to another business, company or entity.

Fig. 2 - JIDEP Language Resource

To this end, the iTelos methodology requires the creation of a language resource associated with the KG, which is created at the end of the whole process. Such a language resource is in the form of a CSV file, which can be created using any spreadsheet tool. Figure 2 shows part of the CSV file defining the language resource

created for the JIDEP project. Each row of the spreadsheet above defines a concept that will be used to build the JIDEP KGs. The language resource columns instead define the main information to be associated with each concept. Specifically, the column set requires the definition of the following information for each concept.

- *Concept ID*: the concept local identifier. This identifier uniquely identifies the respective concept and its meaning every time the concept needs to be used in other data structures (such as datasets or ontologies). The concept ID is not affected by polysemy (different meanings for the same natural language word), which is instead present when the concept is represented by its own natural language term (word).

- *Word*: one of the natural (or domain) language terms that can be used to represent the relative concept.

- *Knowledge Source*: the knowledge source from which the concept was collected, if it was reused from other reference ontologies, or created, if the concept is part of a vocabulary or ontology created specifically for the current KG construction process. The knowledge source value for each concept is the URL of the ontology (or vocabulary) from which the concept was collected. Most of the concepts disambiguated in the JIDEP language resource, partially shown in Figure 2, were created for the JIDEP project and therefore have the JIDEP ontology URL [6] as their knowledge source value. However, some of the JIDEP concepts have been reused from existing standard ontologies in order to improve the interoperability of the final KG. For example, the concepts in rows 4, 6 and 7 in Figure 2 were collected from the EMMO ontology [7], the FOAF ontology [8] and the DBpedia ontology [9], respectively.

- *Knowledge Type*: the type of use for the relative concept. The knowledge type value specifies how the concept is used during the process of building the KG and in the final KG. Such a value defines whether a concept defines an EType (“Class” value in the spreadsheet), an EType data property (“Data Property” value in the spreadsheet), an EType object property (“Object Property” value in the spreadsheet), an entity name (“Entity” value in the spreadsheet) or an entity attribute name (“Attribute” value in the spreadsheet).

- *Description*: the description of the meaning of the concept used for the current purpose. The description value is fundamental for the disambiguation of the relative concept. Thanks to such a description, it is possible to immediately understand the information that can be expressed by using the relative concept in other data structures (such as ontologies and datasets). The description value is extracted directly from the knowledge source indicated in the third column (see Figure 2).

In total, for the JIDEP project, 114 concepts have been collected and disambiguated, thus defining the JIDEP vocabulary, or the JIDEP language resource.

3.4 Schema Alignment Tool

In the fourth iTelos phase, the objective is to generate the schema of the final KG. As already anticipated in section 2.1, such a schema is concretely implemented by an ontology built by reusing as much as possible portions of already existing, well-known reference standard ontologies. In other words, such a requirement needs to be supported by a tool that is able to compose ontologies, as well as to model thus purpose-specific ontological aspects that cannot be reused from existing ontologies. To this end, iTelos provides the usage of the Protègè [2] tool. This tool allows the user to define classes of entities, called Entity Types (or ETypes), by naming them with specific labels and defining the relative properties. The ETypes properties in Protègè are divided into data and object properties. While the former specifies the attributes of a single EType (like name, gender, and date of birth of the Person EType), the latter specifies the relations among different ETypes (for example, the object property “live_in” defines the relation between the EType Person and EType House).

The Protègè tool, in the fourth iTelos phase, takes in input the knowledge resources (reference standard ontologies) collected in the previous phases and the purpose-specific ER model. The objective to achieve by using the tool is to generate a unique ontology, following the ER model, which specifies how the information has to be structured to support the main purpose, and by reusing as much as possible the ETypes and properties of the input reference standard ontologies. The modelling of the final KS’s ontology is implemented by following the knowledge modelling approach defined in D2.3. For this reason, the modelling activity requires a certain level of experience in knowledge management, and, if the iTelos user doesn’t have such kinds of skills, she should be supported by a knowledge engineer. The final output to be produced by the Protègè tool is an RDF-OWL file defining the KG’s unique ontology. It is important to notice how the input knowledge resources, that have been properly formatted in RDF-OWL, reduced the effort in modelling the KG’s unique ontology using Protègè, making it possible to copy and paste the ETypes and properties directly using the tool. Figure 3 depicts a usage example of the Protègè tool.

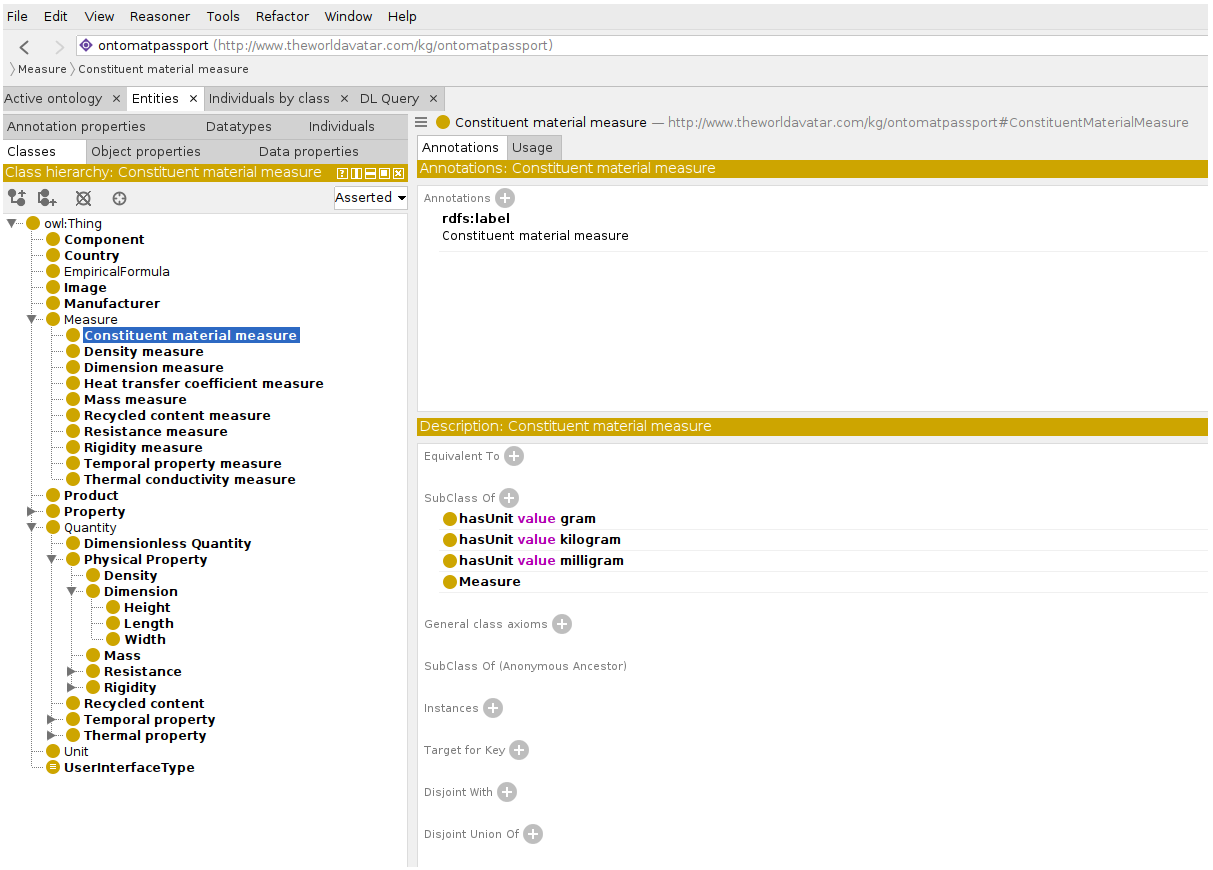


Fig. 3 - Protègè tool over the Material Passport ontology

3.5 Data Mapping Tool

In the last iTelos phase, the objective is the creation of the final KG. To this end, the datasets collected, filtered, cleaned and formalised have to be associated with the unique ontology defined in the previous phase. This activity is called data mapping, namely the mapping of each value in the different datasets, with the ETypes and properties of the unique ontology. The key idea is to represent the information included within all the datasets using a single representation provided by the KG's ontology. To this end, iTelos provides a data mapping tool called Karmalinker, based on the already existing Karma [3] data mapping tool.

The tool takes in input the unique ontology and the set of datasets to be integrated into the final KG. The user will execute the data mapping dataset by dataset, by producing in output a set of RDF files which can be composed together (concretely by unifying the contents) as different parts of the same KG. It is important to notice that the output produced by the Karmalinker tool is a KG where the knowledge and data layer, represented by the datasets and ontologies handled along the iTelos phases, are merged into a single object (RDF file). The Karmlinker tool allows its users to perform two different kinds of operation, described below.

- **Data Mapping Operations**: through this kind of operation, Karmalinker allows the users to declare the association between each dataset field (for example,

table's columns and/or JSON fields) and the EType's properties defined to represent such information. In other words, the mapping operations merge the datasets with the ontology, by representing the entities implicitly defined into the datasets through the ETypes, and their properties, explicitly defined into the unique ontology. Karmalinker allows to perform such operations through a dedicated user interface, where the user can select the right EType and EType's properties for each dataset field. Figure 4 shows an example of the mapping operation between a small automotive test dataset and the automotive ontology defined for the JIDEP use case.

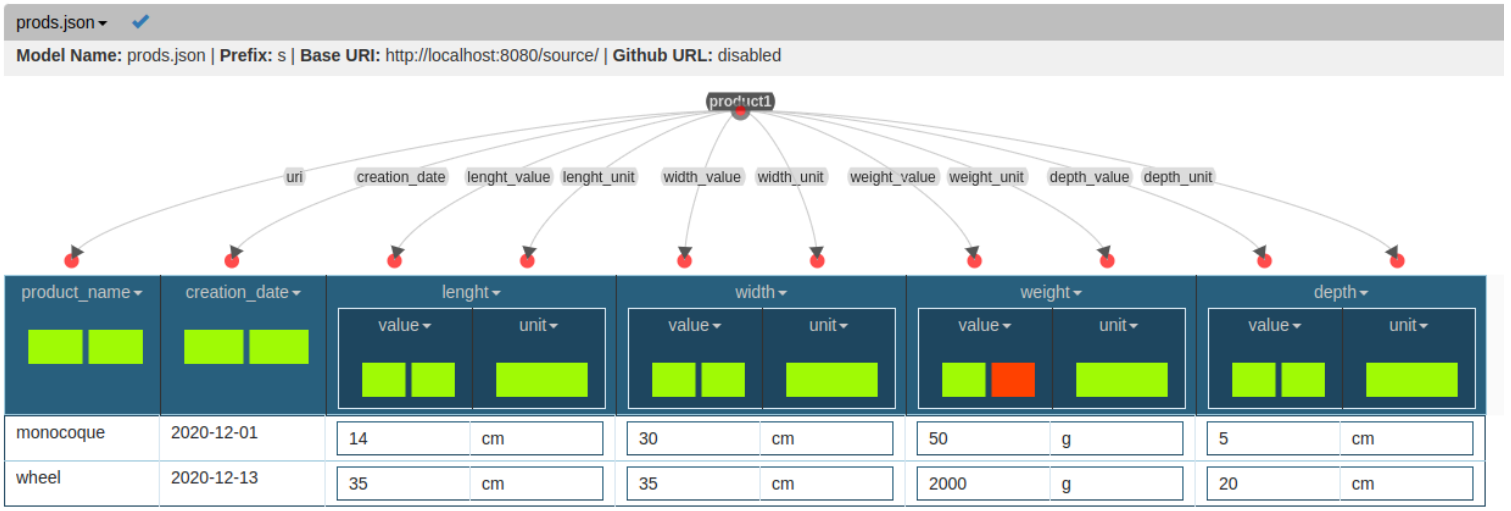


Fig. 4 - Karmalinker data mapping operations.

- **Data Management Operations**: Karmalinker offers another kind of functionalities which allows the user to perform basic data manipulation activities. As depicted in Figure 5, Karmalinker provides the possibility to write Python code to modify the dataset fields. In the example reported in Figure 5, the user exploits the ISODate Python library to format the String date into the standardised ISO format. Different Python libraries can be exploited in Karamlinker to perform data management operations that have not been done in the previous phases. It is important to notice the choice to keep Python as the main programming language for data management across the different methodology phases. This important aspect allows the users to rely on a single library set for this kind of operation, thus limiting the skills required to handle the data, and remaining compatible with the data handled in the previous phases.

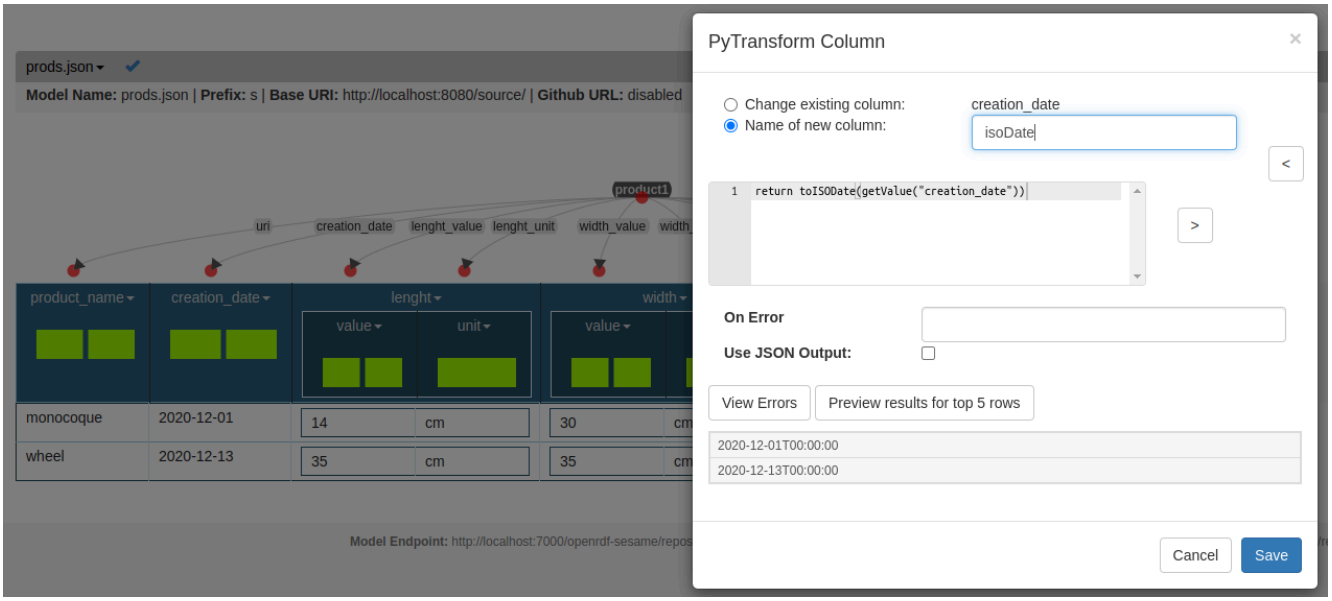


Fig. 5 - Karmalinker data management operations.

As already mentioned, the Karmalinker output is a single RDF-turtle (ttl) where the values in the dataset have been transformed by the data management operations, and associated with the proper EType's properties by the data mapping operations. Karmalinker produces one RDF-turtle file for each dataset handled. The contents of each ttl file can be copied and pasted into a single ttl file, thus composing the final KG. The final KG can be visualised by any tool of Graph and Knowledge visualisation. For example, in Figure 6 an example of KG produced by Karmalinker is visualised in GraphDB [4].

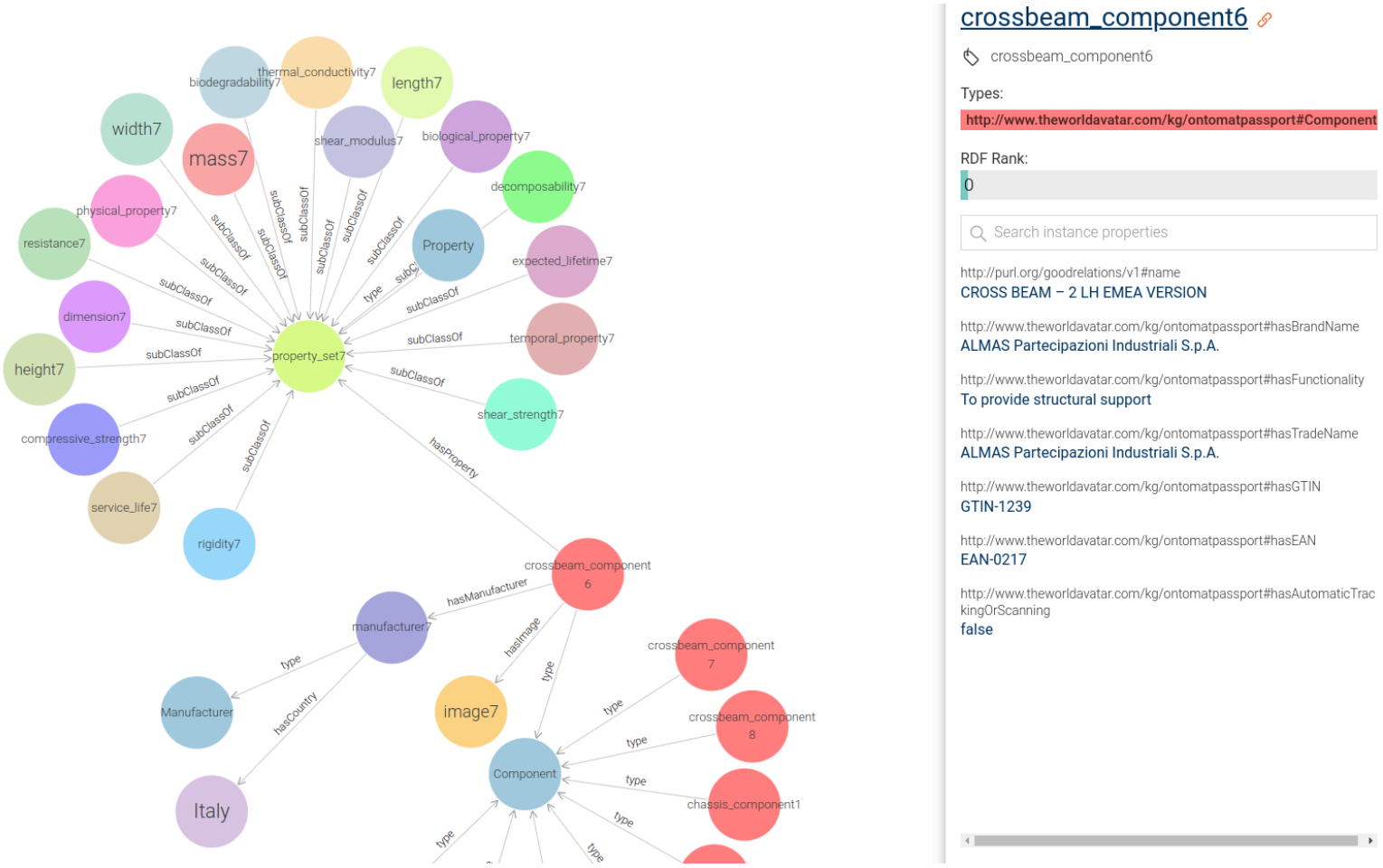


Fig. 6 - Karmalinker KG example.

As described above, Karmalinker offers different functionalities, and some of them can be properly exploited only if a certain level of expertise in data management (and/or Python programming). For this reason, it is strongly suggested the support of a data scientist, during the usage of the tool in the last phase of the methodology.

3.5.1 Karmalinker Semi-automatic Mode

Karmalinker includes another functionality that provides strong support when a huge amount of data having the same structure, or schema, (i.e., datasets having the same fields (dataset schema) but different values) needs to be processed. Such a functionality allows recording into a mapping model file (RDF-turtle) all the mapping operations and data management operations (see Figure 7), performed over one dataset and considering one reference ontology. The mapping model file, which can be produced through the Karmalinker GUI, can be reapplied, considering the same reference ontology, over a different dataset, if and only if the new dataset has the same schema. If that is the case, Karmalinker is able to perform again each operation performed over the previous datasets, automatically, thus significantly

reducing the time required to process the new dataset. Such functionality can be defined as semi-automatic due to the fact that a first execution of Karmalinker is always required to perform and record all the operations required. Nevertheless, after that, all the other executions, over new datasets (having a fixed schema) can be processed automatically by running Karmalinker in the background (for example through the command line).

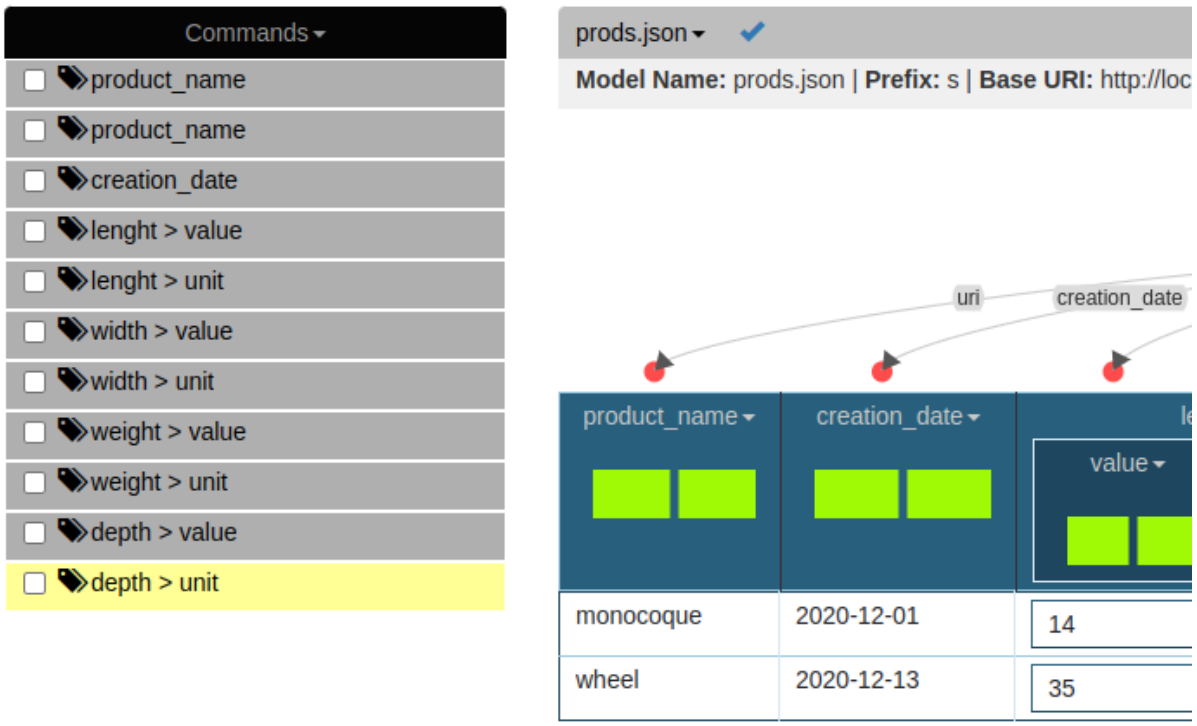


Fig. 7 - Karmalinker, operations recording.

3.5.2 Download and Set-up Instructions

Karmalinker is distributed, under request, by the Knowdive research group, University of Trento (Italy). To achieve the JIDEP project objective, the tool is provided within a dedicated docker image, together with the *docker-compose* file containing the docker objects definition required to execute the tool itself. For this reason, to be able to use the Karmalinker tool, docker (minimum version 2.2), and docker-compose, have to be installed in the deployment machine. To use the Karmalinker tool, it is necessary to store, in a dedicated deployment machine, the relative docker image and *docker-compose* file, and then run the following command:

#docker compose up

If no issues are raised during the installation, the user can access the Karmaliner GUI through a web browser at the following address:

http://localhost:7000/

For the execution of Karmalinker in semi-automatic mode, the user can use the command line (i.e., unix bash shell) by writing the command following the instructions provided on the documentation page [5].

3.6 KG-builder tool

The tool described in this section is a background service for the creation of KGs, to be integrated into the JIDEP DLT platform, in order to transform into KGs the data handled by the platform, by following the iTelos methodology. The tool, called KG-builder, is a software component that includes the execution of Karmalinker in semi-automatic mode, to transform JSON datasets into reusable KGs. The KGs produced by the DLT platform, thanks to the KG-builder tool, will be then published in a dedicated JIDEP data catalogue (see D3.3), where the data can be searched and eventually downloaded for further reuse. Below, in the section, more details are provided about the functionalities of the tool as a standalone service, as well as the integration of the KG-builder tool with the DLT platform.

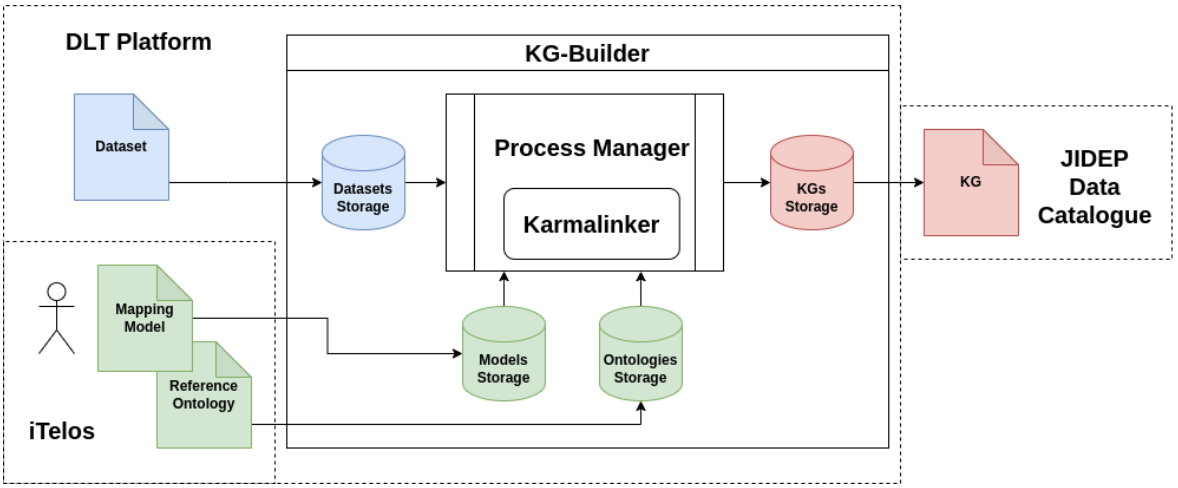


Fig. 8 - KG-builder tool architecture

Figure 8 depicts the architecture of the KG-builder tool, and how it interacts with the DLT platform. The tool takes in input the JSON datasets provided by the DLT platform and aims at providing output a KG-based version of such datasets. The execution of the KG-builder tool is fully automatic once it is deployed within the DLT platform. Nevertheless, as already anticipated, the tool is based on the semi-automatic execution of Karmalinker. Due to that, in order to build the KGs starting from the input datasets, the tool needs the predefined files defining:

- a reference ontology, used to uniquely represent the data of each KG produced. And,
- a mapping model, for each dataset schema, containing the Karmalinker operations to be performed once a dataset having a specific schema has to be transformed into KG.

This means that the KG-builder tool can be exploited after at least one execution of the process that generates a KG (or a portion of it) following the iTelos methodology. Such an iTelos process generates the unique reference ontology and the mapping models that will be used subsequently by the KG-builder tool to transform new datasets (with a recognized schema) in KGs. Notice that, as a consequence, each dataset having a different schema needs its own mapping model, defining the Karmalinker operations to build the KG, specific for that dataset schema. The KG-builder applies the model to the right dataset by matching the name of the two files.

The KG-builder tool is internally composed by a main process (Process Manager in Fig. 7) that handles the collection of the inputs, the matching between datasets and mapping models, as well as the delivery of the final output. Such a process is responsible for the background execution of an instance of the Karmalinker tool that, executed in semi-automatic mode, builds the KGs. The remaining components of the KG-builder tool, are storages, concretely implemented as docker volumes, where the reference ontologies (Schema Storage), the mapping models (Models Storage), the input datasets (Datasets Storage) and the output KGs (KGs Storage) are respectively maintained and handled by the main Process Manager.

3.6.1 Requirements and set up

KG-builder is distributed, under request, by the Knowdive research group, University of Trento (Italy). To achieve the JIDEP project objective, the tool is provided within a dedicated docker image, together with the *docker-compose* file containing the docker containers and services definition required to execute the tool itself. For this reason, to be able to use the KG-builder tool, docker (minimum version 2.2), and docker-compose, have to be installed in the deployment machine. To use the KG-builder tool, it is necessary to store, in a dedicated deployment machine, the relative docker image and *docker-compose* file, and then to run the following command:

```
#docker compose up
```

3.6.2 Usage

The KG-builder is used through its RESTful API layer which consists of two endpoints to be invoked with their respective parameters. The first API endpoint is a POST request, as shown below, which is used to send the dataset to be transformed into a KG.

```
POST:http://<serverIP>:3001/dataset
```

The <serverIP> indicates the IP address of the machine where the tool has been deployed. The body of the first POST request is a JSON format content having two main key-value objects, as shown by the following JSON dataset structure example.

```
{
   "id": <dataset-ID>,
   "passport": <JSON-dataset-content>
{
```

In the JSON dataset structure example above, the `<JSON-dataset-content>` is the content of the dataset that needs to be transformed into a KG. The KG-builder tool will name such a new KG by using the unique identifier provided by the `<dataset-ID>` value. The expected output for the first API request is an acknowledgement message indicating that the tool has successfully received the JSON dataset to be transformed into an interoperable KG.

The second API endpoint is a GET request, shaped as follows, that is used to transform and retrieve the KG version of the dataset sent through the previous request.

GET:http://<serverIP>:3001/build?id=<dataset-ID>&domain=<dataset-domain>

As for the previous request, the <serverIP> indicates the IP address of the machine on which the tool is running. The `<dataset-ID>` in the above request, is exactly the value of the identifier specified in the body of the first request, identifying the dataset to be transformed, that was previously sent to the tool. In addition, the `<dataset-domain>` parameter can take one of the following three values: *automotive*, *pcb* or *wind*. This last parameter defines which of the previously defined data mapping models (see previous section) must be applied to transform the respective dataset into a KG. The expected output for the second API request is the KG version of the previously sent dataset, shaped as an RDF-turtle file, provided directly in the response body.

3.6.3 Deployment and DLT platform integration

This section describes the KG-builder tool deployment as well as its integration into the DLT platform-based service (from now on called “the platform”) for the JIDEP project use cases (see deliverable D3.4 - DLT platform components).

Deployment

More in detail, the deployment environment chosen for the KG-builder tool is a Kubernetes [10] cluster hosted by the DigitalOcean service [11]. The KG-builder deployment process includes the configuration of the Kubernetes cluster through the definition of a manifest file. The YAML manifest file, for the KG-builder tool, defines the state of the tool as it has to be deployed into the Kubernetes cluster. In other words, the tool environment variables, the tool API endpoints, as well as its interaction with other platform services managing the tool’s lifecycle and execution. To better understand the deployment process described in this section, the deliverable reports here below are some technical definitions used to specify the

deployment of the KG-builder tool as it is reported into the manifest YAMAL file (see figure 11).

- *Deployment*: A Kubernetes definition that specifies the desired state for application instances. The Deployment represents the top level state definition into the manifest file used to deploy the KG-builder tool.
- *Service*: A Kubernetes definition that exposes an application running as a network service within a cluster. The Service defines how the KG-builder can be used through its API layer.
- *ConfigMap*: A Kubernetes definition used to store configuration information represented as key-value pairs. A ConfigMap is used to define in the manifest the values of the KG-builder tool parameters. Figure 9 reports an extraction of the manifest information defining a ConfigMap for the KG-builder tool deployment.
- *Secret*: A Kubernetes definition used to store sensible data, such as passwords and API keys, securely. A Secret is defined as part of the manifest to store securely the KG-builder tool information that must not be accessible. Figure 10 reports an extraction of the manifest information defining a Secret for the KG-builder tool deployment.
- *PersistentVolume* (PV): A Kubernetes definition that specifies a request for storage, within the related cluster, used to persist data during, and beyond the KG-builder tool execution.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kg-builder-config
  labels:
    app: kg-builder-api
data:
  REPO_OWNER: repo owner name
  REPO_NAME: repo name
  FILE_PATH: file to change
  ORG_PATH: org path
  COMMITTER_NAME: name of committer
  COMMITTER_EMAIL: email of committer
  COMMIT_MESSAGE_NEW_DATASET: commit msg
  COMMIT_MESSAGE_NEW_ORG: commit org
  COMMIT_BRANCH: commit branch
```

Fig. 9 - KG-builder Deployment ConfigMap

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: kg-builder-secret
  labels:
    app: kg-builder-api
type: Opaque
data:
  ACCESS_TOKEN: <base64 encoded personal access token>
```

Fig. 10 - KG-builder Deployment Secret

The complete information set for the KG-builder tool is defined in the YAML manifest file as it is reported in Figure 11. Once all the deployment state information is properly added into that file, the deployment process can be finalised by executing the following command, in the machine where the Kubernetes cluster is running, to save the manifest file as "*kg-builder-deployment.yaml*" and apply the deployment:

```
#kubectl apply -f kg-builder-deployment.yaml
```

After applying the deployment manifest, it is possible to verify that the deployment is running correctly by checking the deployment status with the following command and checking their output respectively:

```
#kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
catalog-api	1/1	1	1	206d
distributed-storage	1/1	1	1	198d
material-passport	1/1	1	1	206d
mci-api	1/1	1	1	206d
ontology-api	1/1	1	1	206d
platform	1/1	1	1	205d
poe-api	1/1	1	1	205d
storage-api	1/1	1	1	205d
storage-redis	1/1	1	1	214d
kg-builder	1/1	1	1	10d

The availability of the KG-builder tool (shown with the value 1 in the last line of the above command output) indicates the correct execution of the deployment process.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kg-builder
  labels:
    app: kg-builder
spec:
  selector:
    matchLabels:
      app: kg-builder
  template:
```

```
12.     metadata:
13.       labels:
14.         app: kg-builder
15.     spec:
16.       containers:
17.       - name: kg-builder
18.         image: ghcr.io/tvsltd/jidep-kg:master
19.         env:
20.         - name: DATABASE_URL
21.           valueFrom:
22.             configMapKeyRef:
23.               key: DATABASE_URL
24.               name: kg-builder-config
25.         - name: NODE_ENV
26.           valueFrom:
27.             configMapKeyRef:
28.               key: NODE_ENV
29.               name: kg-builder-config
30.         - name: AWS_DEFAULT_REGION
31.           valueFrom:
32.             configMapKeyRef:
33.               key: AWS_DEFAULT_REGION
34.               name: kg-builder-config
35.         - name: LOG_LEVEL
36.           valueFrom:
37.             configMapKeyRef:
38.               key: LOG_LEVEL
39.               name: kg-builder-config
40.         - name: S3_BUCKET_NAME
41.           valueFrom:
42.             configMapKeyRef:
43.               key: S3_BUCKET_NAME
44.               name: kg-builder-config
45.
50.         - name: AWS_ACCESS_KEY_ID
51.           valueFrom:
52.             secretKeyRef:
53.               key: AWS_ACCESS_KEY_ID
54.               name: kg-builder-secret
55.         - name: AWS_SECRET_ACCESS_KEY
56.           valueFrom:
57.             secretKeyRef:
58.               key: AWS_SECRET_ACCESS_KEY
59.               name: kg-builder-secret
60.         resources:
61.           limits:
62.             memory: "256Mi"
63.             cpu: "1000m"
64.         ports:
65.         - containerPort: 3001
66.       imagePullSecrets:
67.         - name: ghcr-secret
```

Fig. 11 - KG-builder Manifest file

Integration to JIDEP Platform:

The KG-builder tool integrates with the JIDEP platform through a well-defined API layer, facilitating seamless communication and data exchange between the tool and the platform's core services. Each API endpoint is designed to handle specific requests and responses, ensuring that data flows efficiently and securely. The interaction process is as follows:

1. **API Requests:** The JIDEP platform sends API requests to the KG-builder tool to send and transform data (see KG-builder tool usage sub section above).
2. **Data Processing:** The KG Builder tool handles the incoming requests internally according to its business logic.The KG Builder tool handles the incoming requests internally according to its business logic.
3. **API Responses:** The results are returned from the tool to the platform via API responses, providing the necessary data or confirmation of actions taken.

4 Conclusions

This deliverable describes how the Knowledge Graph (KG) generation process, defined by the iTelos methodology, is supported by a dedicated framework, composed of different tools. The different tools aim at producing the different intermediate outputs, along the iTelos phases. The key idea, introduced by the iTelos methodology, is to compose the different tools, by considering as input of each tool, the output of the tool executed previously, following the methodology. Such an approach, in the iTelos implementation, aims at reducing the effort in the execution of the different activities required to build a KG. Moreover, the objective is to provide a complete set of tools for KG generation, starting from the formalisation of the requirements, until the generation of a unique object (file or set of files) representing the final KG. The current deliverable reports the description of each tool adopted along the iTelos methodology, as well as how to use them to achieve the desired results.

5 Updates since the last version

This is the second and last version of the current deliverable. The changes that have been applied respect to the previous version are list here below:

- The section 3.3 describing the tool to implement the language alignment activity required by the third phase of the iTelos methodology, has been detailed. In such a section has been reported how to meet the requirements defined by the iTelos methodology, regarding the disambiguation of the terminology used by the KG created.

- The section 3.6 describing the KG-builder tool has been updated by providing more detail about the download, set-up, usage and deployment of the last version of the KG-builder tool.

References

- [1] https://www.yworks.com/products/yed
- [2] https://protege.stanford.edu/
- [3] https://usc-isi-i2.github.io/karma/
- [4] https://www.ontotext.com/products/graphdb/
- [5] https://github.com/usc-isi-i2/Web-Karma/wiki
- [6] http://www.theworldavatar.com/kg/ontomatpassport
- [7] http://emmo.info/emmo
- [8] http://xmlns.com/foaf/0.1
- [9] https://dbpedia.org/ontology
- [10] https://kubernetes.io/
- [11] https://www.digitalocean.com/

Acronyms and Abbreviations

ADL	ALMAS Partecipazioni Industriali S.P.A.
ADS	Adscensus, MB
AVO	Arteevo Technologies Ltd
BUL	Brunel University London
CRF	Centro Ricerche Fiat Scpa
FHV	Fachhochschule Vorarlberg GMBH
PVI	Precision Varionic International Limited
TPI	TPI Composites
TVS	Technovative Solutions Ltd
UCAM	The Chancellor Masters And Scholars Of University Of Cambridge
UNITN	University Degli Studi Di Trento
UPCE	Univerzita of Pardubice
ZOREN	Zorlu Enerji Elektrik Uretim As

CFRP	Carbon Fibre Reinforced Plastic
CO2	Carbon Dioxide
DLT	Distributed Ledger Technology
EC	The European Commission
ELV	End-of-Life-Vehicle
EOL	End-of-Life
GW	Giga-Watt
IC	Integrated Circuit
Mt	Mega-tons
NMF	Non-Metallic Fraction
PCB	Printed Circuit Board
R&D	Research & Development
RSD	Requirements Specification Document
SME	Small-Medium Enterprise
WEEE	Waste Electrical and Electronic Equipment