PROJECT DELIVERABLE REPORT

Grant Agreement Number: 101058732

Joint Industrial Data Exchange Platform

Type: Deliverable Report

D2.6 Report on Ontology and Query APIs integration

Issuing partner	University of Cambridge (UCAM)
Participating partners	Universita di Trento (UNITN) Technovative Solutions (TVS) Brunel University London
Document name and revision	D2.6 Report on Ontology and Query APIs integration
Author	Dr Feroz Farazi
Deliverable due date	31 August 2023
Actual submission date	31 August 2023

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	

Contents

Disclaimer

Acronyms and Abbreviations

ABox	Assertion Box
API	Application Programming Interface
BUL	Brunel University London
DBMS	Database Management System
EMMO	Elementary Multiperspective Material Ontology
JDBC	Java Database Connectivity
JSON-LD	JavaScript Object Notation for Linked Data
LGPL	GNU Lesser General Public License
OSI	Open Source Initiative
OWL	Web Ontology Language
RDF	Resource Description Framework
RDF4J	RDF for Java
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
SWRL	Semantic Web Rule Language
TBox	Terminology Box
Turtle	Terse RDF Triple Language
TVS	Technovative Solutions
UCAM	University of Cambridge
UNITN	University of Trento

Executive Summary

The main goal of this deliverable is to create code that represents top-level, domain-specific, and application-specific ontologies using the Web Ontology Language (OWL). Additionally, we aim to seamlessly integrate these ontologies and APIs into the JIDEP platform for efficient querying. To accomplish this, we developed a tool named TBox Generator. Utilising the OWL API, this tool parses ontological classes and properties from a CSV template, the structure of which was collaboratively defined with relevant project partners. This parsing enables the representation of these elements using OWL. To ensure the accuracy and validity of our ontological model, we applied the Protégé ontology editor. For visualisation, we utilised the OWLViz plugin, an extension for Protégé. The capabilities of the TBox Generator were further expanded to convert OWL representations into JavaScript Object Notation for Linked Data (JSON-LD). We combined the OWL API and Jena into the code to support various query types, including SPARQL queries.

Moreover, we successfully applied the SWRL API, empowering us to define rules for calculating the material circularity index and executing rule-based queries. We collected data from diverse sources, including material passport data provided by project partners and defined the preliminary version of the material passport ontology within the project. This ontology was produced using the TBox Generator and represented in OWL. The data, related metadata, and corresponding ontologies were fed into the iTelos toolset to generate the first version of the JIDEP knowledge graph. This knowledge graph was deployed using the RDF4J server, distributed across multiple nodes to enhance efficiency and reliability. We integrated the RDF4J API into the code to interact with the knowledge graph effectively. The code is integrated into the JIDEP platform, enabling seamless access to ontology and data within the knowledge graph and facilitating operations such as uploading, querying, updating, and deleting.

1. Introduction

The objective of integrating ontology and query APIs into the JIDEP platform involved several activities in the context of the task associated with this deliverable report. These activities aimed at developing code to integrate top-level, domain and application ontologies required for JIDEP into its platform. In achieving the objective, a Comma Separated Values (CSV) template was designed in collaboration with the relevant project partners to define ontological classes, object properties and datatype properties.

In an ontology, the elements known as classes and properties collectively constitute what is referred to as a Terminology Box (TBox). We designed and developed a TBox Generator using Java to parse ontological elements from the CSV template and to represent them using Web Ontology Language (OWL). In its development, the OWL API was used to create an ontological model, axiomatise class, object property and data property representations within the model, and save the model as an OWL file. We collected classes and properties of the preliminary version of the composite material ontology and material passport ontologies from ontology-defining activities performed in T2.3 and T2.4. While the composite material ontology captures the relevant domain knowledge, the material passport ontology describes both the domain and application-specific knowledge. The Elementary Multiperspective Material Ontology (EMMO) was imported to include the coverage of top-level and material domain-specific ontological elements.

The TBox Generator was extended to generate OWL representations in JSON-LD to meet the demand of applications that rely on such representation formalism, potentially enhancing interoperability. The Protégé ontology editor was used to render the OWL and JSON-LD files, view the hierarchical organisation of classes and properties and assess the consistency and validity of ontological axioms. For graphical representation, visualisation and navigation of concepts and their subsumption or IS-A hierarchies, we used the OWLViz plugin for Protégé. The OntoGraf Protégé plugin can also be used as an alternative tool.

Data and metadata collected in T2.1 and the ontologies reused, refined and created in T2.3 and T2.4 were provided as inputs to the iTelos toolset designed and implemented in T3.2. iTelos aligned data and metadata elements with ontological classes and properties, created a mapping file and generated the preliminary version of the JIDEP knowledge graph. The ontologies and knowledge graph were uploaded to the instances of the RDF4J Server deployed on a number of nodes.

The activities also include code development for integrating query APIs into the platform. The OWL API allows the user to perform Description Logic-based queries against ontological representations, and Jena supports SPARQL queries. In creating rules for calculating the material circularity index relying on material passport data of various products collected in T2.1, we used the Semantic Web Rule Language (SWRL) API. This API and the Semantic Query-Enhanced Web Rule Language (SQWRL) API are applied to answer rule-based queries. The RDF4J API enables querying, updating and deleting ontological data from any SPARQL endpoint, providing access to a triple store hosting ontologies and knowledge graphs. We incorporated these APIs in the code and integrated the code for ontology and query APIs into the JIDEP platform.

The deliverable report is structured as follows. Section 2 provides an overview of the state-of-the-art ontology and query APIs for publishing and querying ontological models with the specification of classes, properties and their relationships. Section 3 elaborates on the ontological TBox generator tool developed by UCAM to represent an ontology using OWL by taking input from a CSV file-based template. It briefly describes how to generate ABox for producing a knowledge graph. Section 4 explains how the ontology and query APIs were integrated into the JIDEP platform, and Section 5 concludes the report.

2. Ontology and Query APIs

2.1 OWL API

The OWL API [1][2][3][4] is an Application Programming Interface (API) designed to create ontologies using the W3C recommended Web Ontology Language (OWL) and to operate on them. Since its availability in 2003, it has been revised many times to maintain its alignment with the latest version of OWL. It has the ability to parse and validate ontological models compatible with OWL and OWL 2 profiles, i.e., OWL 2 QL, OWL 2 EL and OWL 2 RL, where QL, EL and RL refer to Query Language, Description Logic EL++ and Rule Language, respectively. It allows serialising OWL ontologies in various syntaxes, including RDF/XML, OWL/XML, Terse RDF Triple Language (Turtle) and JSON Linked Data (JSON-LD) [1].

Java is used as the programming language for implementing the OWL API, which is released as open source under the GNU Lesser General Public License (LGPL). LGPL is a free software license. The OWL API has been included in many projects since its release, and the most renowned ones are Protégé, the Pellet OWL DL reasoner and the HermiT OWL reasoner.

The OWL API's design philosophy is to reduce software developers' specific burdens while interacting with OWL ontologies [1]. Some example interactions are parsing and serialisation of OWL data structures. The OWL API views and manages an ontology as a set of axioms and annotations and as a resource that has an identifier or Internationalised Resource Identifier (IRI) [1][2]. Within the API, an ontology manager creates and manipulates one or more ontologies.

Basic design principles of the OWL API allow access to an ontology via interfaces, editing of the ontology via change operations, the ontology to be independent of concrete serialisations, and separation between components enabling, for example, representation, update, parsing and rendering [1][2]. The OWL API includes a model that allows access to an ontology via interfaces and classes, where the interfaces provide read-only access to the ontology. The model has methods that enable answering questions, such as class containment. For example, the Composite Material class belongs within the Material Passport ontology. While the reference implementation of the OWL API supports efficient in-memory renditions of ontologies, its design has considered the storage of ontologies in databases or triple stores.

Interfaces that are crucial in managing ontologies in the reference implementation of the OWL API are OWLOntology and OWLOntologyManager [1]. The OWLOntology interface enables access to the axioms of an ontology, and it can be implemented differently to configure a different storage medium for ontologies. The OWLOntologyManager interface allows creating and saving an ontology as well as loading and changing the ontology. The OWLOntologyManager interface considers an instance of the OWLOntology interface as an ontology. The OWLReasoner interface provides the reasoning functionality to verify OWL ontologies' consistency and compute transitive closures of class and property hierarchies. Though the reference implementation does not support performing SPARQL queries, the OWLReasoner interface provides essential query support to retrieve instances of classes, child or parent types of a given class, and characteristics of object and datatype properties. This reasoner interface supports the query functionality, relying on its ability to check entailments. Due to the increasing demand for interpretability, OWL API has included a feature to generate explanations for entailments.

The OWL API abstracts away not only from various serialisation syntaxes but also from triples [1]. The abstraction of the API is at the level of ontological axioms, which is considered at a level higher than the level of triples, as each complex axiom corresponds to multiple triples. In OWL 2, axioms serve as the lowest-level data representation units. Therefore, when comparing ontology fragments, it is more sensible to do so at the level of axioms rather than triples.

2.2 Jena API

Jena [5][6] is an API developed by HP Labs using Java to streamline the development of applications that leverage the ontology-centric advanced knowledge model and Semantic Web languages. The core component of Jena is the RDF API that facilitates the creation, update, deletion and query of RDF graphs consisting of nodes and edges. The RDF API provides functionality to store in different technologies. Serialisation of RDF graphs in various knowledge representation languages is supported through plug-ins [5]. Similarly, plug-in support is available to read back RDF graphs from these languages.

The RDF API has two divergent methods to view an RDF graph [5]. One of them is the statement-centric view that considers an RDF graph as a set of triples, each consisting of a source node, a target node and an arc connecting these two nodes. This view proves advantageous in reading, writing or merging multiple graphs collectively. The other is the frame-centric view that resembles the object-oriented programming approach and considers an RDF graph as a group of resources, each consisting of properties. This view proves advantages in graph navigation or operating on a resource alone.

Jena is a free, open-source framework developed using Java and released under the Apache License. It has been used by many organisations and initiatives, for example, the W3C to check the validity of RDF/XML documents, HP Labs in the E-Person project to represent a person as an agent within a network and the Dublin Core Metadata Initiative (DCMI) to create different prototypes for multilingual schema registries [5].

Jena employs the ARP parser to read RDF/XML representations of ontological classes, their instantiations or properties [5]. The syntax processor of ARP is a standard XML processor, as any RDF/XML representation is an XML representation. The Jena API can store RDF/XML data in main memory to meet the demand of applications that do not need large or persistent storage and that require faster query and upload speed. For data-intensive applications, Jena offers a relational database store that can be configured to represent data in any database management system (DBMS) that accepts Java Database Connectivity (JDBC). Jena offers data storage in the Berkley DB for applications needing persistent storage and much faster query execution time but does not require relational database-like transaction support [5].

The query language of the Jena API is RDQL, which is similar to SQL, which stands for Structured Query Language [5][7]. Similarly to SQL, in RDQL, the SELECT clause includes variables defined to extract intended results. The WHERE clause can consist of one or more triples where the subject, predicate, and object can all be variables, matching with all possible triple patterns in RDF graphs or a mix of variables and constants, matching with specific triple patterns. RDQL also supports defining prefixes via the USING clause.

2.3 SWRL API

The Semantic Web Rule Language (SWRL) API is a Java API developed to operate on SWRL rules created on OWL ontologies and to work with the Semantic-Enhanced Web Rule Language (SQWRL) [8]. A SWRL rule is a Horn-like rule consisting of an antecedent (body), a consequent (head) and an implication operating between the antecedent and consequent [9]. Both the antecedent and consequent include a set of conditions. The interpretation of the implication is that if all conditions in the antecedent are met, the conditions provided in the consequent must also be met [8]. SQWRL includes SQL-like operators to retrieve information available in an OWL ontology. The antecedent of an SWRL rule and an SQWRL query are alike. Still, the consequent differs as SQWRL includes operators like “select”, “selectDistinct”, “count”, “countDistinct”, “avg”, “max”, “limit”, and “orderBy” instead of relying solely on conditions. A condition can comprise an ontology class, property, variable, or literal.

Within the SWRL API, the Drools rule engine supports the execution of SWRL rules and SQWRL queries. The SWRL API leverages the OWL API to effectively manage OWL ontologies to associate a rule/query engine for creating rules/queries by utilising ontological classes, properties and instances.

The SWRL API is free and open-source software published under the 2-clause BSD License, one of the licenses approved by the Open Source Initiative. It has been used on the Protégé ontology editor as a plugin, rendered as a feature named SWRLTab. Another use of the API is in developing a standalone application called SWRLTab, equipped with graphical user interfaces to facilitate the editing and execution of SWRL rules and SQWRL queries.

The SWRLTab plugin of Protégé consists of several core components, including the rule and rule-based query editor, rule engine bridge, built-in bridge providing support for mathematical operators, string operators, etc., and a set of Java libraries [10].

The rule editor allows the creation, editing and retrieval of SWRL rules and SQWRL rule-based queries. OWL ontology classes, properties, property values and instances, SWRL and SQWRL specifications and operators, and all XML schema data types can be referred from such rules or queries.

The rule engine bridge invokes a rule reasoning engine to execute SWRL rules/queries defined in the rule layer of an OWL ontological model [10]. Following the creation or edition of a rule, it can be saved in the OWL ontology in target storage. The rule engine bridge provides the functionality to retrieve rules and ontological elements, such as classes and properties, from the ontology to the in-memory structures of the bridge. It can copy the rules and ontology elements to the rule reasoning engine. The bridge facilitates the incorporation of engine-inferred knowledge into the in-memory structures of the bridge. The bridge also provides a feature to write back the in-memory ontology content with inferred knowledge into the target storage of the ontology.

The SWRL built-ins are analogous to functions and can take arguments. The swrlb namespace is designed to use and refer to these built-ins. Some example built-ins are swrlb:lessThan, swrlb:greaterThan, swrlb:add, swrlb:subtract, swrlb:multiply, swrlb:divide, swrlb:substring, and swrlb:stringEqualIgnoreCase. An extended list of built-ins defined for comparisons, mathematical operations, and string operations is provided below, considering their applicability to SWRL rules required for the JIDEP project.

- swrlb:equal – it returns true if the first and second arguments are the same.
- swrlb:notEqual – it returns true if the first argument is not the same as the second argument.
- swrlb:lessThan – it returns true if the first argument is less than the second argument. Otherwise, it returns false.
- swrlb:lessThanOrEqual – it returns true if the first argument is less than or equal to the second argument. Otherwise, it returns false.

- swrlb:greaterThan – it returns true if the first argument is greater than the second argument. Otherwise, it returns false.
- swrlb:greaterThanOrEqual: returns true if the first argument is greater than or equal to the second argument. Otherwise, it returns false.
- swrlb:add – it may have three or more arguments, and the arithmetic sum of the second to the last argument is assigned to the first argument.
- swrlb:subtract – it requires three arguments, and the arithmetic difference between the second and third argument is assigned to the first argument.
- swrlb:multiply – it may have three or more arguments, and the arithmetic product of the second to the last argument is assigned to the first argument.
- swrlb:divide – it requires three arguments, and the result of the second argument divided by the third argument is assigned to the first argument.
- swrlb:pow – it requires three arguments, and the result of raising the second argument to the power of the third argument is assigned to the first argument.
- swrlb:stringEqualIgnoreCase – it returns true if the first argument is the same as the second one in a case-insensitive comparison.
- swrlb:stringConcat – it may have three or more string arguments, and it concatenates all the strings sequentially from the second through the last and assigns the resulting string to the first argument.
- swrlb:substring – it requires four arguments. The second argument contains the input string from which a substring of optional length, as defined in the fourth argument, is extracted starting from the character specified in the third argument. The resulting substring is then assigned to the first argument.
- swrlb:contains – it returns true if the first argument contains the second argument in a case-sensitive matching comparison.
- swrlb:containsIgnoreCase – it returns true if the first argument contains the second argument in a case ignored matching comparison.
- swrlb:startsWith – it returns true if the first argument starts with the string contained in the second argument.
- swrlb:endsWith – it returns true if the first argument ends with the string contained in the second argument.
- swrlb:matches – it returns true if the first argument matches the regular expression in the second argument.

In addition to these built-ins, SWRLAPI provides a built-in bridge that enables users to define and implement custom built-ins using Java, integrating user-defined built-ins seamlessly with the existing ones [11][12][13].

2.4 RDF4J API

RDF for Java (RDF4J) [14] is a framework developed to operate on RDF, assisting programmers and developers in creating applications that leverage the power of the Semantic Web and Linked Data, published by applying rules proposed by Tim Berners-Lee. The RDF4J framework provides a triple store and query engine for RDF and OWL data. RDF4J includes an API called RDF4J API, a REST API designed and implemented to support SPARQL 1.1 Protocol[1] compliant query and update operations, and method of processing requests, performing these operations and returning results via HTTP to the requesting system [15].

The RDF4J API can establish links and perform query and update operations against core RDFJ databases, i.e., Memory Store, Native Store and Elasticsearch Store, as well as third-

[1] https://www.w3.org/TR/sparql11-protocol/

party RDF4J-compatible database solutions, such as Blazegraph, GraphDB, Stardog, Amazon Neptune, Oracle RDF Graph Adapter, and Openlink Virtuoso [16]. These databases are also known as triple stores. RDF4J supports many serialisation formats, including RDF/XML, JSON-LD, N-Triples, and N-Quads.

The Repository API [17] is central to accessing RDF4J-compliant triple stores and SPARQL endpoints to query or update data and knowledge. Within the Repository API, the core interface is Repository, which has been implemented for various purposes. One such implementation is the HTTPRepository, developed as a bridge between a repository created on a remote RDF4J Server and a client. Another implementation is SPARQLRepository, which bridges a remotely deployed SPARQL endpoint and a client application. Another such implementation is SailRepository, allowing the access and creation of a local RDF4J repository. SailRepository can be configured differently to create a repository to store RDF data in memory or disk.

RDF data stored in an RDF or OWL file can be manually added to an RDF4J repository through the RDF4J Workbench User Interface (UI) and the Repository API programmatically by indicating the path to the file. An instance of RepositoryConnection is created, and the add operation is executed to upload the file. A context may be included while adding the file, allowing its download and deletion.

Some Semantic Web applications may require multiple repositories, and managing static references to individual repositories in different parts of code is cumbersome [17]. RepositoryManager has been designed and developed to address this issue. LocalRepositoryManager is one implementation of RepositoryManager to manage local repositories, and RemoteRepositoryManager is for managing remote repositories.

The RDF4J Workbench User Interface has provided the option to build a federation of repositories that can be queried via a single endpoint. However, such federations suffer from performance issues while dealing with tens of thousands of triples in each repository. In addition to the federation of RDF4J repositories, the framework also has provided a query feature using FedX to perform a federated query against multiple endpoints [18]. The performance of this latter approach is acceptable as it returns results at a reasonable time when queries are performed to retrieve data and knowledge from DBpedia and Wikidata endpoints that host massive amounts of data.

The RDF4J framework, RDF4J API and Repositories are licensed under the Eclipse Distribution License (EDL), version 1.0, which is one of the licenses approved by the Open Source Initiative (OSI). RDF4J and its API have been used by many organisations and Communities, including the British Broadcasting Corporation (BBC), OpenLink Software, Ontotext and the Semantic Web and Linked Data communities.

3. Ontology Generator

Within an ontology, classes, properties and their relationships are called TBox, and instances of classes, relations between instances and datatype properties with associated data are called ABox.

3.1 Ontological TBox Generator

The Ontological TBox Generator is developed by UCAM using the OWL API to represent classes and properties of a TBox provided in a JIDEP consortium agreed Comma-Separated Values (CSV) file-based template in the Web Ontology Language (OWL). The TBox Generator requires a description of the metadata of a TBox, including the IRI, version, and comment. Users can utilise a text editor such as Notepad++ or spreadsheet software like Google Sheets or Microsoft Excel to fill out the CSV file-based template. The following subsections describe filling out the template using a text editor.

3.1.1 CSV file-based template

3.1.1.1 Header Row

The header row consists of the following attributes or columns:

```
Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
```

1. Source: the ontological element that needs to be defined.
2. Type: the type of the element provided in the Source column.
3. Target: the element that is related to the element in the Source column.
4. Relation: the relationship between the element in the Source column and the element in the Target column.
5. Domain: the domain of an object property or a data type property.
6. Range: the range of an object property or a data type property.
7. Quantifier: the quantifier that imposes a restriction on an object property or data type property.
8. Comment: a natural language description of the element provided in the Source column.
9. Defined By: the IRI of the ontology that defines the element in the Source column.
10. Label: a natural language label or name of the Source column element.

3.1.1.2 Value Rows

Values rows include TBox metadata, classes, object properties and data properties.

3.1.1.2.1 TBox Metadata

1. The value rows that MUST follow the header row in the template are the TBox IRI, version, comment and import rows.
2. Assume that you want to develop a TBox called OntoMatPassport with the IRI http://www.theworldavatar.com/kg/ontomatpassport/, version number 1 and the comment "OntoMatPassport is an ontology defined for representing product, component and material passports" and by importing the Elementary Multiperspective Material Ontology (EMMO) TBox that has the IRI https://raw.githubusercontent.com/emmo-repo/EMMO/1.0.0-beta5/emmo.ttl, then fill out the template as follows.

```
Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
```
```
OntoMatPassport,TBox,http://www.theworldavatar.com/kg/ontomatpassport/, https://www.w3.org/2007/05/powder-s#hasIRI,,,,,,
```

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
OntoMatPassport,TBox,1,http://www.w3.org/2002/07/owl#versionInfo,,,,,,
OntoMatPassport,TBox, OntoMatPassport is an ontology developed for representing product, component and material passports,http://www.w3.org/2000/01/rdf-schema#comment,,,,,,
OntoMatPassport,TBox, https://raw.githubusercontent.com/emmo-repo/EMMO/1.0.0-beta5/emmo.ttl,http://www.w3.org/2002/07/owl#imports,,,,,,

3.1.1.2.2 Classes

Ontological classes MUST follow the TBox Data and Metadata block. Assume that in JIDEP, we want to:

1. Define classes Product, Component, Material, Manufacturer, Property, Physical Property and Composition Property with the following descriptions:
   1.1 Product: An artefact manufactured, possibly in an industrial setting, to meet the demands of consumers and usually consists of multiple parts.
   1.2 Component: A product usually part of a bigger whole that can be a system or a product of composite nature.
   1.3 Material: A product of a specific type or quality that is a tangible substance or matter.
   1.4 Manufacturer: A business, company or entity manufacturing goods for the consumer market.
   1.5 Property: A characteristic of an object that can be applied in determining its use in applications.
   1.6 Physical Property: A property to express the physical aspect of an object.
   1.7 Composition Property: A property to describe the composition aspect of an object.
2. Describe that Component and Material are subclasses of the Product class, and Physical Property and Composition Property are subclasses of the Property class. The Material class is equivalent to the EMMO_4207e895_8b83_4318_996a_72cfb32acd94 class defined in the Elementary Multiperspective Material Ontology (EMMO) ontology available at https://raw.githubusercontent.com/emmo-repo/EMMO/1.0.0-beta5/emmo.ttl.
3. Specify that these classes are defined in an ontology with the following URL: https://raw.githubusercontent.com/cambridge-cares/TheWorldAvatar/dev-composite-materials-ontology/JPS_Ontology/ontology/ontomatpassport/ontomatpassport.owl.

To achieve these goals, we have filled out the template as follows.

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
Product,Class,,,,,,An artefact manufactured possibly in an industrial setting to meet the demands of consumers and usually consists of multiple parts.,http://www.theworldavatar.com/kg/ontomatpassport,Product

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
Component,Class,Product,IS-A, , , ,A product usually part of a bigger whole that can be a system or a product of composite nature.,http://www.theworldavatar.com/kg/ontomatpassport,Component
Material,Class,Product,IS-A,,,,A product of a specific type or quality that is a tangible substance or matter.,http://www.theworldavatar.com/kg/ontomatpassport,Material
Material,Class,http://emmo.info/emmo#EMMO_4207e895_8b83_4318_996a_72cfb32acd94,EQUIVALENT-TO,,,,A product of a specific type or quality that is a tangible substance or matter.,http://www.theworldavatar.com/kg/ontomatpassport,Material
Manufacturer,Class,,,,,,A business company or entity manufacturing goods for the consumer market.,http://www.theworldavatar.com/kg/ontomatpassport,Manufacturer
Property,Class,,,,,,A characteristic of an object that can be applied in determining its use in applications.,http://www.theworldavatar.com/kg/ontomatpassport,Property
PhysicalProperty,Class,Property,IS-A,,,,A property to express the physical aspect of an object.,http://www.theworldavatar.com/kg/ontomatpassport,Physical Property
CompositionProperty,Class,Property,IS-A,,,,A property to describe the composition aspect of an object.,http://www.theworldavatar.com/kg/ontomatpassport,Composition Property

3.1.1.2.3 Class-class relationships

The OWL language-supported constructs rdfs:subClassOf and owl:equivalentClass are used for representing subclass of, and equivalent class relationships, respectively. The mapping between these relationships and the TBox Generator acceptable relationships is shown below:

OWL relationship	TBox Generator acceptable relationship
rdfs:subClassOf	IS-A
owl:equivalentClass	EQUIVALENT-TO

3.1.1.2.4 Object Properties

Object properties can be represented just below TBox Metadata rows. However, it is recommended that object properties should be provided after classes. Assume that in JIDEP we need to:

1. define the object properties hasManufacturer, hasImage and hasProperty with the following descriptions:

1.1 hasManufacturer: An object property referring to the manufacturer of a component orproduct.
1.2 hasImage: An object property referring to the image of a product or component.
1.3 hasProperty: An object property referring to different properties of a component or product.

2. specify that a Product has a Manufacturer, a Product has an Image and a Product has a Property.

To achieve these goals, we have filled out the template as follows:

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
hasManufacturer,Object Property,,,Product,Manufacturer, ,An object property referring to the manufacturer of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Manufacturer
hasImage,Object Property,,,Product,Image, ,An object property referring to the image of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Image
hasProperty,Object Property,,,Product,Property,,An object property referring to different properties of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Property

3.1.1.2.5 Datatype Properties

Datatype properties or data properties can be represented just below TBox Metadata rows. However, it is recommended that data properties are provided after classes. Data properties can be provided above or below object properties.

Assume that you want to:

1. define the data properties id, name, EAN, tradeName and registrationNumber with the following descriptions:

   1.1 id: An alphanumeric string that identifies a product or component.
   1.2 name: A linguistic identity of a product, component or material.
   1.3 EAN: A datatype property expressing the European Article Number (EAN) of a product or component.
   1.4 tradeName: A name that indicates the business or company name.
   1.5 registrationNumber: A number given to a manufacturer when registration is completed with a regulatory body.

2. specify that the data type of these properties is String

To achieve these goals, we have filled out the template as follows:

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
id, Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Id

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
name,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Name
EAN,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,EAN
tradeName,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Trade name
registrationNumber,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Registration number

3.1.2 Conversion into OWL and JSON-LD

To demonstrate the conversion, we copied the TBox metadata and the partial list of classes and properties of the Material Passport ontology provided in the CSV file-based template in Section 4.1.1, pasted them into a newly opened Notepad++ file, removed empty lines and saved the file as a CSV file. The complete content of this CSV file is shown in Appendix A. After creating the CSV file, we converted it into OWL by running the TBox Generator. The generated OWL representation is shown in Appendix B. As shown in Figure 1, we visualised the editable ontology version in Protégé. A Java library was used to convert the OWL example representation into JSON-LD, as shown in Appendix C.

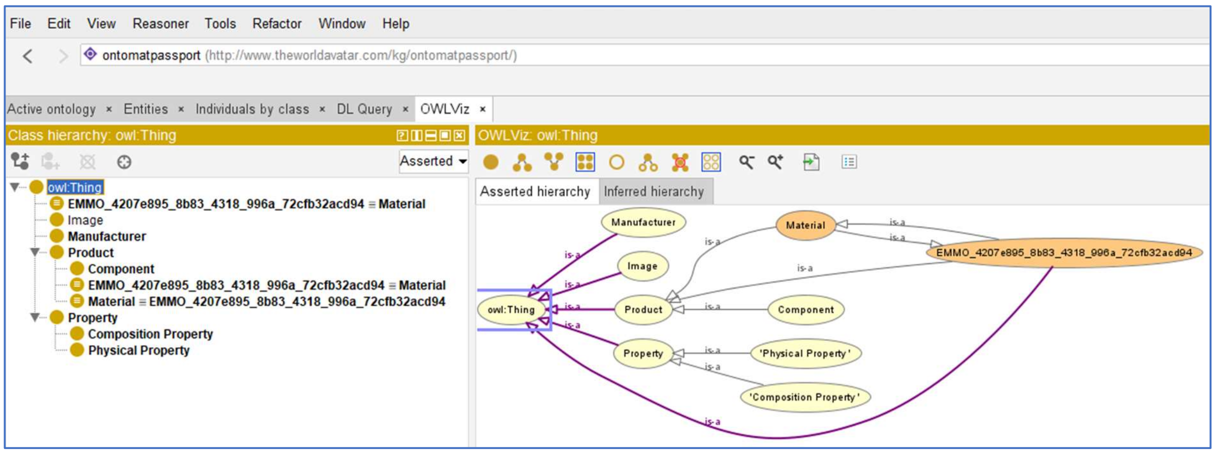

Figure 1. Screenshot of a portion of the material passport ontology, visualised in Protégé, generated by the TBox Generator using OWL APIs. This version of visualisation was produced by removing the reference to the EMMO ontology.

3.2 Ontological ABox Generator

Data and metadata collected from the data-providing partners were available in JSON format. The iTelos toolset, developed by UNITN in T3.2, aligned the data and metadata with the first version of the Material Passport ontology, produced a mapping and generated a knowledge graph with data and metadata instantiated in RDF, a format interoperable with OWL.

4. API Integration

We used, tested and analysed the OWL API, Jena, and SWRL API to understand their support for ontology generation and query features. We adopted a wrapper-based approach to abstract these APIs' complexity and facilitate seamless integration with the JIDEP platform. The OWL API was integrated into the code developed in T2.6 to enable users and applications to perform Description Logic-like queries, while the integration of Jena allowed for issuing SPARQL queries. The integration of the SWRL API facilitated the creation of rules for various calculations, including material circularity, and the execution of rule-based SQWRL queries.

We installed the RDF4J server on multiple nodes, and on these nodes, we deployed the knowledge graph generated using the iTelos toolset. We also integrated the RDF4J API in the code to support creating, querying, updating, and deleting operations on the knowledge graph. These operations are performed in a way that maintains a consistent and coherent copy of the knowledge graph distributed across these nodes.

We integrated the code into the JIDEP platform in collaboration with TVS. The APIs integrated into the code will not only support the core operations of the knowledge graph but also enable seamless access to all JIDEP ontologies from the platform, as well as data, metadata, and knowledge represented using these ontologies within the knowledge graph. This comprehensive integration enhances the accessibility and utility of the knowledge graph, enabling effective utilisation of the underlying data and knowledge resources.

5. Conclusions

Integrating ontology and query APIs into the JIDEP platform involved various steps. It included creating code to incorporate different ontologies into the platform, designing a CSV template to define ontological elements, and building a TBox Generator using Java to represent these elements using OWL. This tool was extended to convert OWL to JSON-LD for enhanced application interoperability.

The Protégé ontology editor was used to validate JIDEP ontologies. The iTelos toolset aligned data with ontological classes, generating a preliminary version of the JIDEP knowledge graph. Query APIs were also integrated into the code developed in this task with OWL API for Description Logic-based queries, Jena for SPARQL queries, and SWRL and SQWRL APIs for rule-based queries. The RDF4J API facilitated querying and managing ontological data. The final step involved incorporating the developed code into the JIDEP platform.

References

[1] Horridge, M. and Bechhofer, S., 2011. The owl api: A java api for owl ontologies. *Semantic web*, *2*(1), pp.11-21.
[2] Horridge, M. and Bechhofer, S., 2009, October. The OWL API: A Java API for Working with OWL 2 Ontologies. In *OWLED* (Vol. 529, pp. 11-21).
[3] Bechhofer, S., Volz, R. and Lord, P., 2003, October. Cooking the Semantic Web with the OWL API. In *International Semantic Web Conference* (pp. 659-675). Berlin, Heidelberg: Springer Berlin Heidelberg.
[4] Horridge, M., Bechhofer, S. and Noppens, O., 2007, June. Igniting the OWL 1.1 touch paper: The OWL API. In *OWLED* (Vol. 258, pp. 6-7).
[5] McBride, B., 2002. Jena: A semantic web toolkit. *IEEE Internet computing*, *6*(6), pp.55-59.
[6] McBride, B., 2001, May. Jena: Implementing the RDF Model and Syntax Specification. In *SemWeb* (Vol. 1, pp. 23-28).

[7] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. and Wilkinson, K., 2004, May. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 74-83).
[8] SWRLAPI. URL https://github.com/protegeproject/swrlapi. Accessed 16 August 2023.
[9] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. URL https://www.w3.org/Submission/SWRL. Accessed 16 August 2023.
[10] O'Connor, M., Tu, S., Nyulas, C., Das, A. and Musen, M., 2007. Querying the semantic web with SWRL. In *Advances in Rule Interchange and Applications: International Symposium, RuleML 2007, Orlando, Florida, October 25-26, 2007. Proceedings 1* (pp. 155-159). Springer Berlin Heidelberg.
[11] O'Connor, M.J., Shankar, R.D., Musen, M.A., Das, A.K. and Nyulas, C., 2008, October. The SWRLAPI: A Development Environment for Working with SWRL Rules. In *OWLED*.
[12] O'connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W. and Musen, M., 2005. Supporting rule system interoperability on the semantic web with SWRL. In *The Semantic Web–ISWC 2005: 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005. Proceedings 4* (pp. 974-986). Springer Berlin Heidelberg.
[13] O'connor, M., Knublauch, H., Tu, S. and Musen, M., 2005. Writing rules for the semantic web using SWRL and Jess. *Protégé With Rules WS, Madrid*.
[14] Eclipse RDF4J. URL https://rdf4j.org/. Accessed 21 August 2023.
[15] RDF4J Rest API. URL https://rdf4j.org/documentation/reference/rest-api/. Accessed 21 August 2023.
[16] The Eclipse RDF4J Framework. URL https://rdf4j.org/about/. Accessed 21 August 2023.
[17] The Repository API. URL https://rdf4j.org/documentation/programming/repository/. Accessed 21 August 2023.
[18] Federation with FedX. URL https://rdf4j.org/documentation/programming/federation/. Accessed 21 August 2023.

Appendix A: Filled in TBox CSV file-based template

TBox metadata, classes, class-class relationship, object properties and datatype properties provided in the TBox CSV file-based template in Section 4.1.1 are combined to produce a CSV file that has the following content, and that can be converted to an OWL ontology using the TBox Generator.

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
OntoMatPassport,TBox,http://www.theworldavatar.com/kg/ontomatpassport/, https://www.w3.org/2007/05/powder-s#hasIRI,,,,,,
OntoMatPassport,TBox,1,http://www.w3.org/2002/07/owl#versionInfo,,,,,,
OntoMatPassport,TBox, OntoMatPassport is an ontology developed for representing product, component and material passports,http://www.w3.org/2000/01/rdf-schema#comment,,,,,,
OntoMatPassport,TBox, https://raw.githubusercontent.com/emmo-repo/EMMO/1.0.0-beta5/emmo.ttl,http://www.w3.org/2002/07/owl#imports,,,,,,
Product,Class,,,,,,An artefact manufactured possibly in an industrial setting to meet the demands of consumers and usually consists of multiple parts.,http://www.theworldavatar.com/kg/ontomatpassport,Product
Component,Class,Product,IS-A, , , ,A product usually part of a bigger whole that can be a system or a product of composite nature.,http://www.theworldavatar.com/kg/ontomatpassport,Component
Material,Class,Product,IS-A,,,,A product of a specific type or quality that is a tangible substance or matter.,http://www.theworldavatar.com/kg/ontomatpassport,Material
Material,Class,http://emmo.info/emmo#EMMO_4207e895_8b83_4318_996a_72cfb32acd94,EQUIVALENT-TO,,,,A product of a specific type or quality that is a tangible substance or matter.,http://www.theworldavatar.com/kg/ontomatpassport,Material
Manufacturer,Class,,,,,,A business company or entity manufacturing goods for the consumer market.,http://www.theworldavatar.com/kg/ontomatpassport,Manufacturer
Property,Class,,,,,,A characteristic of an object that can be applied in determining its use in applications.,http://www.theworldavatar.com/kg/ontomatpassport,Property
PhysicalProperty,Class,Property,IS-A,,,,A property to express the physical aspect of an object.,http://www.theworldavatar.com/kg/ontomatpassport,Physical Property
CompositionProperty,Class,Property,IS-A,,,,A property to describe the composition aspect of an object.,http://www.theworldavatar.com/kg/ontomatpassport,Composition Property
hasManufacturer,Object Property,,,Product,Manufacturer, ,An object property referring to the manufacturer of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Manufacturer
hasImage,Object Property,,,Product,Image, ,An object property referring to the image of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Image

Source,Type,Target,Relation,Domain,Range,Quantifier,Comment,Defined By,Label
hasProperty,Object Property,,,Product,Property,,An object property referring to different properties of a product.,http://www.theworldavatar.com/kg/ontomatpassport,Property
id, Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Id
name,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Name
EAN,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,EAN
tradeName,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Trade name
registrationNumber,Data Property,,,,string,,,http://www.theworldavatar.com/kg/ontomatpassport,Registration number

Appendix B: Generated OWL ontology

The ontological elements included in the CSV file in Appendix A are converted into OWL using the TBox Generator, and the resulting OWL representation is shown below.

```
<?xml version="1.0"?>

<rdf:RDF xmlns="http://www.theworldavatar.com/kg/ontomatpassport/"

     xml:base="http://www.theworldavatar.com/kg/ontomatpassport/"

     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

     xmlns:owl="http://www.w3.org/2002/07/owl#"

     xmlns:xml="http://www.w3.org/XML/1998/namespace"

     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

     xmlns:ontomatpassport="http://www.theworldavatar.com/kg/ontomatpassport/"

     xmlns:dc="http://purl.org/dc/elements/1.1/">

    <owl:Ontology rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/">

        <owl:imports rdf:resource="https://raw.githubusercontent.com/emmo-repo/EMMO/1.0.0-beta5/emmo.ttl"/>

        <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#string">23 August 2023</dc:date>

        <gitCommitHash
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">c0599beca8df55873a1ab061dee64e52c510c6a0
</gitCommitHash>

        <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1</owl:versionInfo>

    </owl:Ontology>

    <!--

    ///////////////////////////////////////////////////////////////////////////////////////

    //

    // Object Properties

    //

    ///////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/hasImage -->

    <owl:ObjectProperty rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/hasImage">

        <rdfs:domain rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Product"/>

        <rdfs:range rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Image"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> An object property referring
to the image of a product.</rdfs:comment>
```

```
        <rdfs:isDefinedBy
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://www.theworldavatar.com/kg/ontomatpasspo
rt</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Image</rdfs:label>

    </owl:ObjectProperty>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/hasManufacturer -->

    <owl:ObjectProperty rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/hasManufacturer">

        <rdfs:domain rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Product"/>

        <rdfs:range rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Manufacturer"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> An object property referring
to the manufacturer of a product.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Manufacturer</rdfs:label>

    </owl:ObjectProperty>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/hasProperty -->

    <owl:ObjectProperty rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/hasProperty">

        <rdfs:domain rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Product"/>

        <rdfs:range rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Property"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> An object property referring
to different properties of a product.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Property</rdfs:label>

    </owl:ObjectProperty>

    <!--

    ///////////////////////////////////////////////////////////////////////////////////////

    //

    // Classes

    //

    ///////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://emmo.info/emmo#EMMO_4207e895_8b83_4318_996a_72cfb32acd94 -->

    <owl:Class rdf:about="http://emmo.info/emmo#EMMO_4207e895_8b83_4318_996a_72cfb32acd94">

        <owl:equivalentClass rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Material"/>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Component -->
    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Component">

        <rdfs:subClassOf rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Product"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A product usually part of a
bigger whole that can be a system or a product of composite nature.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Component</rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/CompositionProperty -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/CompositionProperty">

        <rdfs:subClassOf rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Property"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> A property to describe the
composition aspect of an object.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Composition Property
</rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Image -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Image"/>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Manufacturer -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Manufacturer">

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> A business company or
entity manufacturing goods for the consumer market.</rdfs:comment>

        <rdfs:isDefinedBy
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://www.theworldavatar.com/kg/ontomatpasspo
rt</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Manufacturer</rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Material -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Material">

        <rdfs:subClassOf rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Product"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> A product of a specific type
or quality that is a tangible substance or matter.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:isDefinedBy
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://www.theworldavatar.com/kg/ontomatpasspo
rt</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Material</rdfs:label>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Material</rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/PhysicalProperty -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/PhysicalProperty">

        <rdfs:subClassOf rdf:resource="http://www.theworldavatar.com/kg/ontomatpassport/Property"/>

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> A property to express the
physical aspect of an object.</rdfs:comment>

        <rdfs:isDefinedBy
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://www.theworldavatar.com/kg/ontomatpasspo
rt</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Physical Property </rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Product -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Product">

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> An artefact manufactured
possibly in an industrial setting to meet the demands of consumers and usually consists of multiple
parts.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Product</rdfs:label>

    </owl:Class>

    <!-- http://www.theworldavatar.com/kg/ontomatpassport/Property -->

    <owl:Class rdf:about="http://www.theworldavatar.com/kg/ontomatpassport/Property">

        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> A characteristic of an object
that can be applied in determining its use in applications.</rdfs:comment>

        <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
http://www.theworldavatar.com/kg/ontomatpassport</rdfs:isDefinedBy>

        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Property</rdfs:label>

    </owl:Class>

</rdf:RDF>

<!-- Generated by the OWL API (version 5.1.0.2017-03-29T23:31:42Z) https://github.com/owlcs/owlapi/ -->
```

Appendix C: Ontological Representation in JSON-LD

The OWL API-generated ontology shown in Appendix B has been converted into JSON-LD using a library. JSON-LD is interoperable with OWL; the content represented using JSON-LD is provided below.

```
[ {
  "@id" : "http://emmo.info/emmo#EMMO_4207e895_8b83_4318_996a_72cfb32acd94",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2002/07/owl#equivalentClass" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Material"
  } ]
}, {
  "@id" : "http://purl.org/dc/elements/1.1/date",
  "@type" : [ "http://www.w3.org/2002/07/owl#AnnotationProperty" ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/",
  "@type" : [ "http://www.w3.org/2002/07/owl#Ontology" ],
  "http://purl.org/dc/elements/1.1/date" : [ {
    "@value" : "24 August 2023"
  } ],
  "http://www.theworldavatar.com/kg/ontomatpassport/gitCommitHash" : [ {
    "@value" : "c0599beca8df55873a1ab061dee64e52c510c6a0"
  } ],
  "http://www.w3.org/2002/07/owl#versionInfo" : [ {
    "@value" : "1"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Component",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A product usually part of a bigger whole that can be a system or a product of composite nature."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Component"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#subClassOf" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/CompositionProperty",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A property to describe the composition aspect of an object."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Composition Property "
  } ],
  "http://www.w3.org/2000/01/rdf-schema#subClassOf" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Property"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/EAN",
  "@type" : [ "http://www.w3.org/2002/07/owl#DatatypeProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "EAN"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Image",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ]
```

```
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Manufacturer",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A business company or entity manufacturing goods for the consumer market."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Manufacturer"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Material",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A product of a specific type or quality that is a tangible substance or matter."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Material"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#subClassOf" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/PhysicalProperty",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A property to express the physical aspect of an object."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Physical Property "
  } ],
  "http://www.w3.org/2000/01/rdf-schema#subClassOf" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Property"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "An artefact manufactured possibly in an industrial setting to meet the demands of consumers
and usually consists of multiple parts."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Product"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Property",
  "@type" : [ "http://www.w3.org/2002/07/owl#Class" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "A characteristic of an object that can be applied in determining its use in applications."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Property"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/gitCommitHash",
  "@type" : [ "http://www.w3.org/2002/07/owl#AnnotationProperty" ]
```

```
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/hasImage",
  "@type" : [ "http://www.w3.org/2002/07/owl#ObjectProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "An object property referring to the image of a product."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#domain" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Image"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Image"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/hasManufacturer",
  "@type" : [ "http://www.w3.org/2002/07/owl#ObjectProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "An object property referring to the manufacturer of a product."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#domain" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Manufacturer"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Manufacturer"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/hasProperty",
  "@type" : [ "http://www.w3.org/2002/07/owl#ObjectProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#comment" : [ {
    "@value" : "An object property referring to different properties of a product."
  } ],
  "http://www.w3.org/2000/01/rdf-schema#domain" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Product"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Property"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/Property"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/id",
  "@type" : [ "http://www.w3.org/2002/07/owl#DatatypeProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Id"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/name",
  "@type" : [ "http://www.w3.org/2002/07/owl#DatatypeProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Name"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/registrationNumber",
  "@type" : [ "http://www.w3.org/2002/07/owl#DatatypeProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Registration number"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
}, {
  "@id" : "http://www.theworldavatar.com/kg/ontomatpassport/tradeName",
  "@type" : [ "http://www.w3.org/2002/07/owl#DatatypeProperty" ],
  "http://www.w3.org/2000/01/rdf-schema#isDefinedBy" : [ {
    "@value" : "http://www.theworldavatar.com/kg/ontomatpassport"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#label" : [ {
    "@value" : "Trade name"
  } ],
  "http://www.w3.org/2000/01/rdf-schema#range" : [ {
    "@id" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
} ]
```