Collaborative Network for Industry, Manufacturing, Business and Logistics in Europe

D3.2

Catalogue Ingestion and Semantic Annotation

Project Acronym	NIMBLE	
Project Title	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe	
Project Number	723810	
Work Package	WP3	Core Business Services for the NIMBLE Platform
Lead Beneficiary	SRDC	
Editor	Suat Gönül	SRDC
Reviewers	Benny Mandler, Dietmar Glachs, Gianluca D'Acosta, Wernher Behrendt	IBM, SRFG, ENEA
Contributors		
Dissemination Level	PU	
Contractual Delivery Date	30/06/2017	
Actual Delivery Date	07/07/2017	
Version	1.0	

Abstract

This document provides details about product catalogue publishing that is the way for companies to make themselves discoverable on NIMBLE. Aiming for a better search experience for users, we introduce semantic annotation for enriching an object of interest for NIMBLE in terms of machine-processable metadata. We follow a similar methodology used by existing eCommerce platforms and let users select a product category so that the Catalogue Registry can recommend product specific metadata during the publishing process. We give a short state of the art of on product taxonomies and explain our reasoning for selecting eClass as the built-in taxonomy of NIMBLE.

We use Universal Business Language (UBL) as the common data model of the Catalogue Registry because it contains appropriate data elements for catalogue/product management that are also connected to broader data elements related to supply chain operations such ordering, fulfilment, delivery and invoicing. We present how we modified the initial UBL schema by cropping several irrelevant data elements and by adding some new elements in response to the data representation requirements of our use cases.

Inspired by the ebXML Registry standard, which provides a specification for sharing of data and metadata between organizational entities in a federated environment, NIMBLE makes a distinction between repository and registry concepts in which actual data and metadata are stored. Currently, we use a global metadata registry, which is populated with metadata that could be obtained from catalogues in varying formats and stored in disparate databases accordingly. We introduce an adapter concept called *Feature Extractor,* which runs on dedicated catalogue / product definition formats and which extracts semantic metadata from the data represented in those formats and populate the metadata registry. The metadata holds references to the repositories where the actual data items are stored.

NIMBLE in a Nutshell

NIMBLE is the collaboration Network for Industry, Manufacturing, Business and Logistics in Europe. It will develop the infrastructure for a cloud-based, Industry 4.0, Internet-of-Things-enabled B2B platform on which European manufacturing firms can register, publish machine-readable catalogues for products and services, search for suitable supply chain partners, negotiate contracts and supply logistics. Participating companies can establish private and secure B2B and M2M information exchange channels to optimise business work flows. The infrastructure will be developed as open source software under an Apache-type, permissive license. The governance model is a federation of platforms for multi-sided trade, with mandatory interoperation functions and optional added-value business functions that can be provided by third parties. This will foster the growth of a net-centric business ecosystem for sustainable innovation and fair competition as envisaged by the Digital Agenda 2020. Prospective NIMBLE providers can take the open source infrastructure and bundle it with sectorial, regional or functional added value services and launch a new platform in the federation. The project started in October 2016 and will last for 36 months.

Document History

Version	**Date**	**Comments**
V0.1	05/06/2017	Initial structure
V0.2	20/06/2017	Input for Furniture Ontology
V0.3	20/06/2017	Input for Catalogue Ontology
V0.4	29/06/2017	Pre-final version for review
V0.5	30/06/2017	Updates based on feedbacks of Dietmar Glachs, Benny Mandler and Gianluca d'Acosta
V0.6	02/07/2017	Final Review: Wernher Behrendt
V1.0	07/07/2017	Submission of final version

Table of Contents

List of Figures

List of Tables

Acronyms

Table 1: Acronyms table

Acronym	**Meaning**
ABIE	Aggregate business information entity
API	Application Programming Interface
B2B	Business to business
BBIE	Basic business information entity
BFO	Basic Formal Ontology
CRUD	Create – read – update – delete
CSV	Comma separated value
ebXML	Electronic Business XML
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JSON	Java Script Object Notation
LDPath	Linked data path
M2M	Machine to machine
NIMBLE	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe
RDF	Resource description framework
REST	Representational State Transfer
SDK	Software Development Kit
SPARQL	SPARQL Protocol and RDF Query Language
UBL	Universal Business Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Introduction

The ultimate aim of NIMBLE is to optimize companies' B2B operations throughout the supply chain by orchestrating and automating data exchange among the participants of the chain in different phases such as sub-contracted manufacturing, transportation, etc. The orchestration of data exchange towards that automation is realized through business processes involving at least two parties with specific roles feeding the process with relevant data.

Partner discovery is the initial step prior to performing B2B operations on NIMBLE unless the user, initiating a business process, already knows which company to look for. Catalogue Registry is the main enabler of the partner discovery phase as it enables companies to introduce themselves to the NIMBLE platform with the products they supply and the services they provide.

In order to enable users to find what they are looking for quickly, Catalogue Registry offers publishing products with semantically relevant annotations. The registry makes use of generic and sector-specific taxonomies as knowledge bases from which relevant annotations can be obtained automatically given a particular product category. This document presents our approach aiming to ease the process for users to publish their products where the publishing process is supported with these taxonomies. We give details about their usage from the users' perspective.

We manage data and metadata of products in different ways. While metadata are kept in a global registry; raw data, which could have varying formats, are kept in disparate repositories. Maintaining all the metadata in a single repository enables querying on products having heterogeneous structures initially. Once a product is identified, its complete, structured definition can be fetched from the respective repository.

Based on the technical design considerations for dealing with the heterogeneity of catalogue / product data formats along with the storage mechanisms for management of data and metadata separately, we provide details about the current status of the API and relevant implementation. We intend to maintain this document beyond the current state, for documentation of future improvements of the Catalogue Registry component.

Not only physical goods can be the subject of a business process but also intangible services like painting, transportation, etc. We will be using the term "product" in the rest of the document referring to both of these concepts.

2 Semantic Annotation

Describing a product with a paragraph of text is not a mechanism on which user-friendly, effective search mechanisms can be built apart from keyword-based search. In contrast to a purely text-based approach, semantic annotation is the method for describing products with a set of machine-readable properties in addition to the free text descriptions. Such annotations can easily be processed, indexed and used for providing advanced search capabilities. They make the discovery process faster and more effective through the use of faceted search, which is the de facto search method used by eCommerce platforms. Utilizing the relations among the product categories as well as relations with other products set via the properties, NIMBLE also provides semantic search on the persisted data.

Semantic search provides several ways of discovery. For example, users can perform exploratory search by broadening or narrowing the search scope by using the categories residing in varying level of hierarchies as filtering criteria. Considering the logistics service hierarchy depicted in Figure 1, users can broaden the search scope by selecting a higher-level category e.g. *logistics* and narrow it by selecting a lower-level category e.g. *train transport*. Furthermore, specific properties of categories can be used to narrow the search scope further e.g. q*uality level* property of *train transport* category. Structured metadata enables widening the search scope with linguistic extensions as well e.g. synonyms of original category names. From the opposite point of view, such additional information can be indexed as metadata of products, and afterwards the products could be discovered for any keyword matching with the indexed metadata. Continuing from the same example, searching with the *logistics* keyword might yield as result the *rail transport* service.

2.1 Product Category Taxonomies

As already stated, product category taxonomies are knowledge bases structured as a hierarchy of categories. In addition to enriching product descriptions with machine processable semantic metadata, such taxonomies provide a common terminology for search processes where single concepts can be represented in multiple languages.

We analyzed a set of available category taxonomies in order to identify the most appropriate one to be used as the default taxonomy in NIMBLE. [1] provides the following table about the coverage of the available taxonomies called classification systems. As can be seen from the table, eClass is the taxonomy with the highest number of classes (i.e. categories) and properties.

Table 2 - Statistics about taxonomies [1]

Classification system	Number of					Class distr. (%)
	levels	classes	properties	individuals	top-level c.	
CPC Ver.2	5	4,409	0	0	10	18
CPA 2008	6	5,429	0	0	21	53
CPV 2008	4	10,419	0	0	254	6
eCl@ss 5.1.4	4	30,329	7,136	4,720	25	18
eCl@ss 6.1	4	32,795	9,910	7,531	27	16
ETIM 4.0	2	2,213	6,346	7,001	54	8
FreeClass 2012	4	2,838	174	1,423	11	21
GPC 2012	4	3,831	1,710	9,562	37	17
proficl@ss 4.0	≤ 6	4,617	4,243	6,815	17	36
WZ 2008	5	1,835	0	0	21	33
Google prod. tax.	≤ 7	5,508	0	0	21	17
productpilot	≤ 8	7,970	0	0	20	28
BMEcat	na	na	0	0	na	na

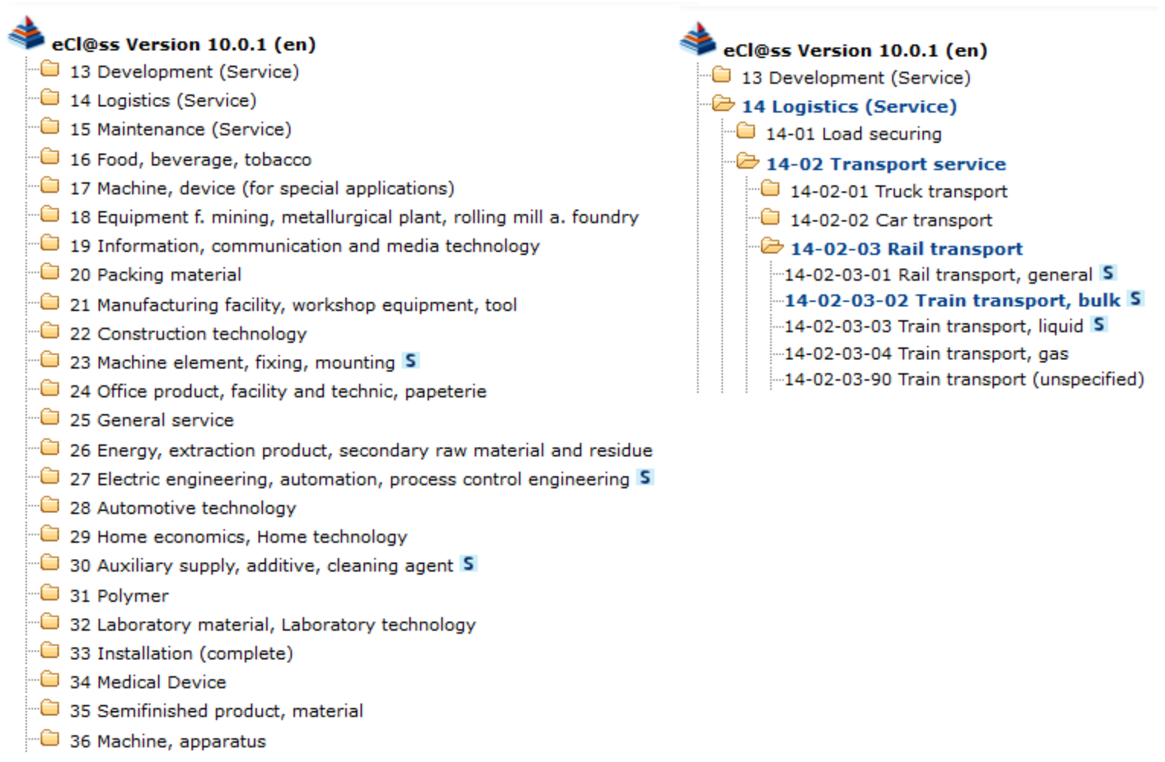
As shown in Annex A - Excerpt of Entities Involved in the MICUNA Use Case, we also compared the taxonomies with respect to the coverage of the concepts used in the Micuna use case. Again, eClass outperforms the other taxonomies. As a result, we decided to use eClass for the generic NIMBLE instance. In the subsequent sections, we provide details about eClass and the Furniture taxonomy, which is the first sector-specific taxonomy that we have integrated into NIMBLE so far.

2.1.1 eClass

eClass is an ISO/IEC compliant industry standard for cross-industry product and service classification. Although we use its English version, it is an international standard with translations to several other languages. The latest version (10.0.1) of the taxonomy was released in February 2017 with 41,000 product categories and 17,000 properties[1]. In line with NIMBLE's objectives, the eClass taxonomy contains resources for both tangible products like furniture fittings and intangible services like rail transportation. The complete hierarchy can be browsed at their website[2].

[1] http://wiki.eclass.eu/wiki/Category:Products

[2] http://www.eclasscontent.com/

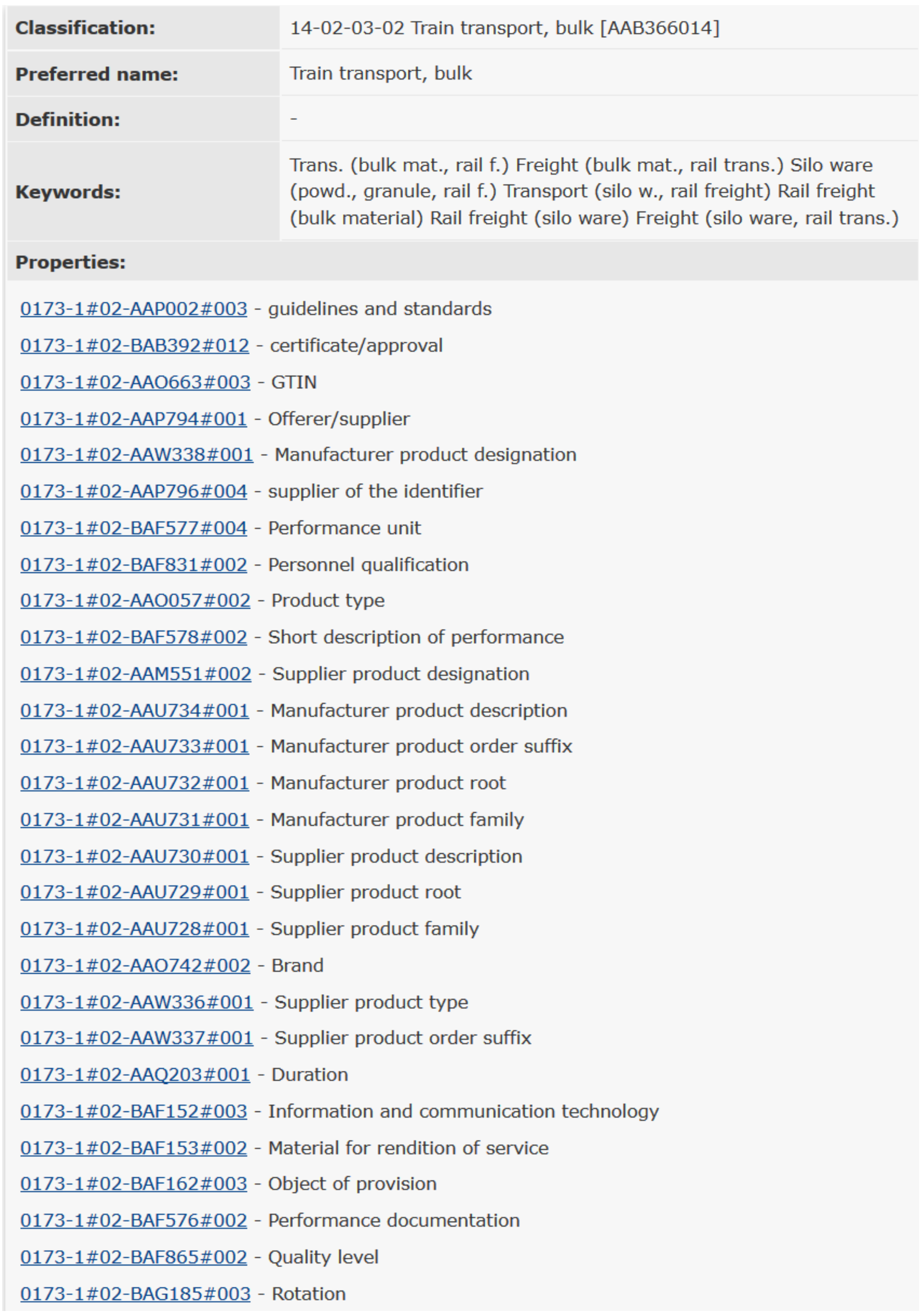


eCl@ss Version 10.0.1 (en)
- 13 Development (Service)
- 14 Logistics (Service)
- 15 Maintenance (Service)
- 16 Food, beverage, tobacco
- 17 Machine, device (for special applications)
- 18 Equipment f. mining, metallurgical plant, rolling mill a. foundry
- 19 Information, communication and media technology
- 20 Packing material
- 21 Manufacturing facility, workshop equipment, tool
- 22 Construction technology
- 23 Machine element, fixing, mounting S
- 24 Office product, facility and technic, papeterie
- 25 General service
- 26 Energy, extraction product, secondary raw material and residue
- 27 Electric engineering, automation, process control engineering S
- 28 Automotive technology
- 29 Home economics, Home technology
- 30 Auxiliary supply, additive, cleaning agent S
- 31 Polymer
- 32 Laboratory material, Laboratory technology
- 33 Installation (complete)
- 34 Medical Device
- 35 Semifinished product, material
- 36 Machine, apparatus

eCl@ss Version 10.0.1 (en)
- 13 Development (Service)
- **14 Logistics (Service)**
  - 14-01 Load securing
  - **14-02 Transport service**
    - 14-02-01 Truck transport
    - 14-02-02 Car transport
    - **14-02-03 Rail transport**
      - 14-02-03-01 Rail transport, general S
      - **14-02-03-02 Train transport, bulk** S
      - 14-02-03-03 Train transport, liquid S
      - 14-02-03-04 Train transport, gas
      - 14-02-03-90 Train transport (unspecified)

**Figure 1 – (Left) Some of eClass root categories,
(Right) Category hierarchy for logistics services**

eClass has a category hierarchy with 4 levels. Figure 1 shows the root level categories on the left; and on the right is the expanded sub-hierarchy for the root logistics category. Figure 2 shows the properties associated to *train transport* category along with some related keywords.



Classification:	14-02-03-02 Train transport, bulk [AAB366014]
Preferred name:	Train transport, bulk
Definition:	-
Keywords:	Trans. (bulk mat., rail f.) Freight (bulk mat., rail trans.) Silo ware (powd., granule, rail f.) Transport (silo w., rail freight) Rail freight (bulk material) Rail freight (silo ware) Freight (silo ware, rail trans.)

Properties:

0173-1#02-AAP002#003 - guidelines and standards
0173-1#02-BAB392#012 - certificate/approval
0173-1#02-AAO663#003 - GTIN
0173-1#02-AAP794#001 - Offerer/supplier
0173-1#02-AAW338#001 - Manufacturer product designation
0173-1#02-AAP796#004 - supplier of the identifier
0173-1#02-BAF577#004 - Performance unit
0173-1#02-BAF831#002 - Personnel qualification
0173-1#02-AAO057#002 - Product type
0173-1#02-BAF578#002 - Short description of performance
0173-1#02-AAM551#002 - Supplier product designation
0173-1#02-AAU734#001 - Manufacturer product description
0173-1#02-AAU733#001 - Manufacturer product order suffix
0173-1#02-AAU732#001 - Manufacturer product root
0173-1#02-AAU731#001 - Manufacturer product family
0173-1#02-AAU730#001 - Supplier product description
0173-1#02-AAU729#001 - Supplier product root
0173-1#02-AAU728#001 - Supplier product family
0173-1#02-AAO742#002 - Brand
0173-1#02-AAW336#001 - Supplier product type
0173-1#02-AAW337#001 - Supplier product order suffix
0173-1#02-AAQ203#001 - Duration
0173-1#02-BAF152#003 - Information and communication technology
0173-1#02-BAF153#002 - Material for rendition of service
0173-1#02-BAF162#003 - Object of provision
0173-1#02-BAF576#002 - Performance documentation
0173-1#02-BAF865#002 - Quality level
0173-1#02-BAG185#003 - Rotation

Figure 2 Properties associated with the "Train transport" category

As shown in Figure 3, only leaf level categories are associated with properties. eClass also constraints the value sets for some properties. Furthermore, some of the properties might have different set of value sets based on the category they are assigned to.

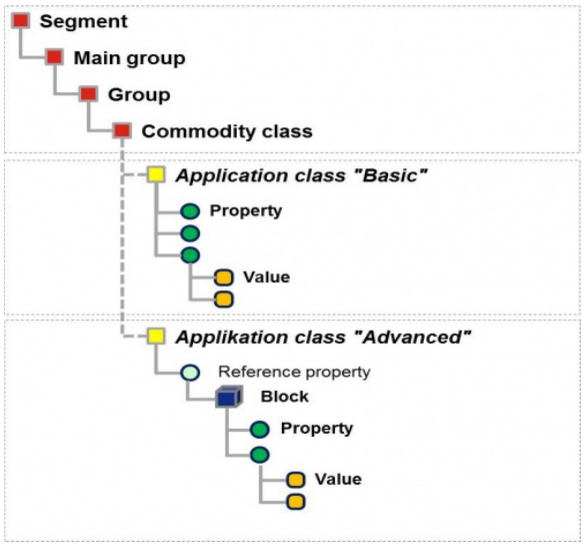


Figure 3 - Structural elements of eClass[3]

We requested eClass resources, a set of CSV files, from the organization maintaining it and transformed them into a machine processable format and persisted it in a relational database in the first version. In more recent versions, we have also kept the taxonomy in a triple store Marmotta[4], which is an open source Apache project. The reason for doing so is to better utilize the semantic relations among the categories of the taxonomy.

2.1.2 Domain-Specific Taxonomies

One of the ways in which NIMBLE can be specialized for specific sectors is to let the sector players be visible on NIMBLE with their sector-specific, structured information. Thanks to the extensible mechanism for additional classification taxonomies, companies can annotate their products with the information available in these taxonomies. It is worth highlighting that an extension of NIMBLE is not a work to be performed by the parties trading on the platform but is an option for platform providers serving NIMBLE to a specific sector. It is also possible that third parties would offer such stronger semantic services on a platform.

2.1.2.1 Furniture Taxonomy

The furniture taxonomy has been updated from a previous version of the taxonomy of processes in the furniture industry and the furniture ontology, both managed by AIDIMME.

As a result, the merged taxonomy includes the following main resources focused on the furniture sector:

[3] http://wiki.eclass.eu/wiki/Category:Structure_and_structural_elements

[4] http://marmotta.apache.org/

- **Processes and techniques**: activities organised by main production stages, from first wood processing until product delivery, as well as related techniques.
- **Machinery**: machines for furniture manufacturing including manual, automatic and semiautomatic devices.
- **Equipment**: tools, protective and safety equipment used by workers.
- **Materials, substances and textiles**: categorisation of most used components in furniture manufacturing.
- **Assembly hardware**: fitting elements used in pieces of furniture.
- **Product catalogue**: categorisation of furniture products considering finished pieces of furniture and furniture complements.

The product catalogue branch has its origins in the Furniture Ontology which is based on the international standard for data exchange ISO 10303-236[5] (Industrial automation systems and integration -- Product data representation and exchange -- Part 236: Application protocol: Furniture catalogue and interior design) also known as funStep, focused on the furniture and wood cluster. This was developed to define a common vocabulary expressed in a formal way and capturing the most used concepts inside the furniture and furniture-related industry, including properties and relationships between them.

2.1.2.2 Textile Taxonomy

The textile taxonomy has been derived from the ontology based on the eBIZ/Moda-ML business standard, which is generated by the CEN initiative.

The eBIZ/Moda-ML standard is mainly focused on the collaboration aspect of the textile/clothing production and covers the processes that involve two or more companies or actors. For this reason, the elements that can be derived from this standard are more related to these collaboration aspects than the classification/categorization of goods.

Thus, the OntoMODA ontology, derived from the standard, is mainly based on the description and categorization of the different business processes, and is poor on the categorization of the products. However, the richness of different types of data in the Moda-ML dictionary related to textile/clothing products, can be used to further develop the catalogue by including also the categorization of goods. Hence, a taxonomy of concepts can be derived from the following resources:

- **Business Processes**: activities organised by different collaboration step linked with relevant documents for the specific interaction;
- **Product catalogues**: collection of yarn, fabric, garment and accessory products.

[5] https://www.iso.org/standard/42340.html

At the moment, the catalogues are plain lists of goods (for example a list of yarns) with different properties (both commercial information and technical datasheet relevant for all the commercial aspects). These properties are described in a very detailed way and the analysis process has taken into account almost all the properties that characterize a textile/clothing product.

On the other hand, a real product catalogue in the textile/clothing sector is a very complex object containing, for example, not only the images of the yarns, but also real samples that differ in colour or in production processes. In the textile/clothing sector, buyers want to hold the product in their hands, look at the colour in a very precise way and select the goods directly.

For this reason, the use of the eBIZ/Moda-ML catalogues has been restricted to a few cases for demonstrational purposes.

However, the realization of electronic textile catalogues for yarn and fabrics may be boosted by the producers in a future NIMBLE exploitation.

2.2 Data Representation

Management of catalogue data is a challenging task in NIMBLE. Firstly, NIMBLE has to deal with on the one hand static data about companies and on the other hand relatively dynamic data for the products they provide and also different trading conditions related to the products. All this information must be kept in a flexible way enabling the development of envisioned search capabilities.

2.2.1 Basic Elements

We start with an overview of the most important conceptual elements in the domain of product catalogues. A high-level visualization of the ontology schema is presented in section 2.2.3 Ontology Schema.

The core aspects within the product catalogue domain are Business Entity, Resource Entity and Dependent Entity. When there is a need for more detailed domain specific descriptions for the Resource Entities, the core concepts can be extended with additional taxonomies such as the eClass taxonomy or furniture domain specific taxonomy using linked data principles. The basic relationship between the entities is depicted in Figure 4. A **business entity** owns **resource entities**, and can offer some of the resource entities to others. The business entity as well as the resource entities, are described in the **dependent entit**y with properties such as contact and price. In addition, the resource entities can be detailed and further specified with **extensions**. In the following, these four concept categories are described:

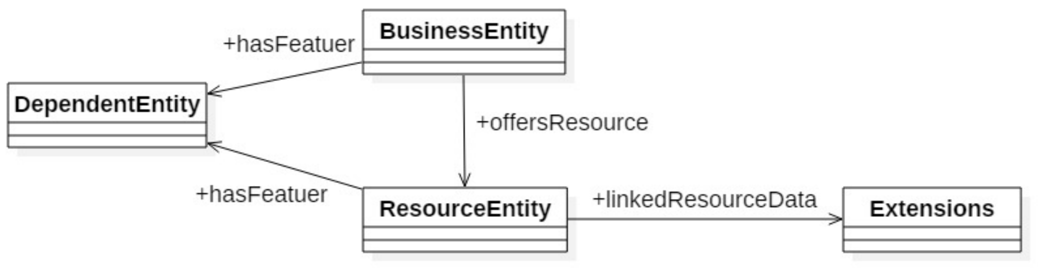


Figure 4 Relationship between main concept categories

Business Entity Concept

A *Business Entity* is a legal agent that participates in some activities in a supply chain. It can be a legal *Party* or a *Person,* which has a naturally detailed description such as name and contacts. A *Business Entity* can offer some *Resources*, or take part in sub-processes in a supply chain, for example, for resources acquisition. When a *Business Entity* performs some activities in a supply chain, it takes usually a *Business Role.* Typical *Business Roles* in a supply chain are *End User*, *Manufacturer, Supplier, Retailer* etc.

Resource Entity Concept

A *Resource Entity* is a kind of product or service that a *Business Entity* holds. It can be an *Item* or *Aggregated Resource* such as *Catalogue.* The entity *Item* is arranged based on the taxonomies in the Basic Formal Ontology (BFO) [2]: *Continuant Resource Entity*, *Occurrent Resource Entity*. *Continuant Resource Entity* is a kind of resource that continues to exist through time, for example, some *Physical Product* or *Digital Resources*. By contrast, *Occurrent Resource Entity* is a kind of resource that exists for a certain time period, such as a *Logistic Service*. Each *Resource Entity* has some resource specific technical characteristics or commercial properties such as price and specification. These kinds of specifications are detailed in the *Dependent Entity* description*.*

Dependent Entity Concept

The next major category of concepts is the *Dependent Entity,* which is mainly dependent upon the *Business Entity* and *Resource Entity*. The concepts in the *Dependent Entity* include entities that are used to describe the details of the *Business Entity* and *Resource Entity*. The concepts dependent upon *Business Entity* can be for example *Contact*, *Financial Account,* and *Business Role.* The concepts for specifying a *Resource Entity* can be for example *Price*, *Commodity Classification*, *Identifier* etc. Based on the concept *Commodity Classification,* a *Resource Entity* can be linked to other extensions such as eClass[6] taxonomy or other product ontologies.

[6] http://www.eclasscontent.com/index.php?language=en

Extensions

The *Extension* is a very special category of concepts used in the NIMBLE platform in order to keep extensibility and usability of the data model for diverse use cases. In different use cases or supply chains, diverse resources are exchanged. Each resource has its own domain specific properties. It is typically hard to manage every single property for all different resources in various domains in a single ontology. There are usually domain specific product ontologies for the description of resources in a specific domain. In the NIMBLE platform, we intend to reuse this kind of domain specific resources to detail the *Resource Entity*. For example, the eClass taxonomy can be applied to classify a *Resource Entity,* and the respective properties in an eClass category are able to specify the *Resource Entity*. In the same way, other product domain taxonomies such as the Furniture Taxonomy in the furniture domain or MODA-ML ontology in the textile domain can be extended. Details on the extension will be given in deliverable D2.2 (Semantic Modelling of Manufacturing Collaboration Assets).

2.2.2 Universal Business Language (UBL)

Universal Business Language (UBL) is a world-wide standard providing a royalty-free library of standard electronic XML business documents that are commonly used in supply chain operations. UBL also defines common business processes as well-defined data exchange flows among trading partners. Cataloguing is one of these processes and UBL defines a document schema[7] for the representation of catalogues.

We use Universal Business Language (UBL) as the common data model of our Catalogue Registry as it contains appropriate data elements for catalogue/product management that are also connected to broader data elements related to supply chain operations such ordering, fulfilment, delivery and invoicing. UBL has good coverage in terms of the data elements required in our use cases. We mapped the data elements required in the Micuna use case to data elements defined in the UBL data model. As can be seen in Annex B, which shows the initial mapping effort, most of the Micuna data elements had corresponding elements in the UBL data model.

In addition to document schemas that are rather complex, UBL provides a set of reusable building blocks for constructing these documents. UBL provides two main groups of reusable elements. Each element in the first group is named a Basic *Business Information Entity (BBIE).* As the name implies such elements can only have one value with varying data types. While data types can be built-in data types such as integer, string, date and so on; they can also be associated with additional attributes such as the measurement unit for a quantity, mime type for a binary object, etc.

[7] http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html#T-CATALOGUE

Entities in the second group are called *Aggregate Business Information Entities (ABIE)*. These are collections of BBIEs and can also have references to other ABIEs. Having specific semantics, *Document* schemas like *Catalogue* have the same structure with ABIEs.

The UBL data model includes main concepts that are required in the scope of the Catalogue Registry component. In addition to the high-level *Catalogue* document concept, it includes concepts for representing companies, persons, catalogues, products, product properties, delivery terms, trading terms and so on.

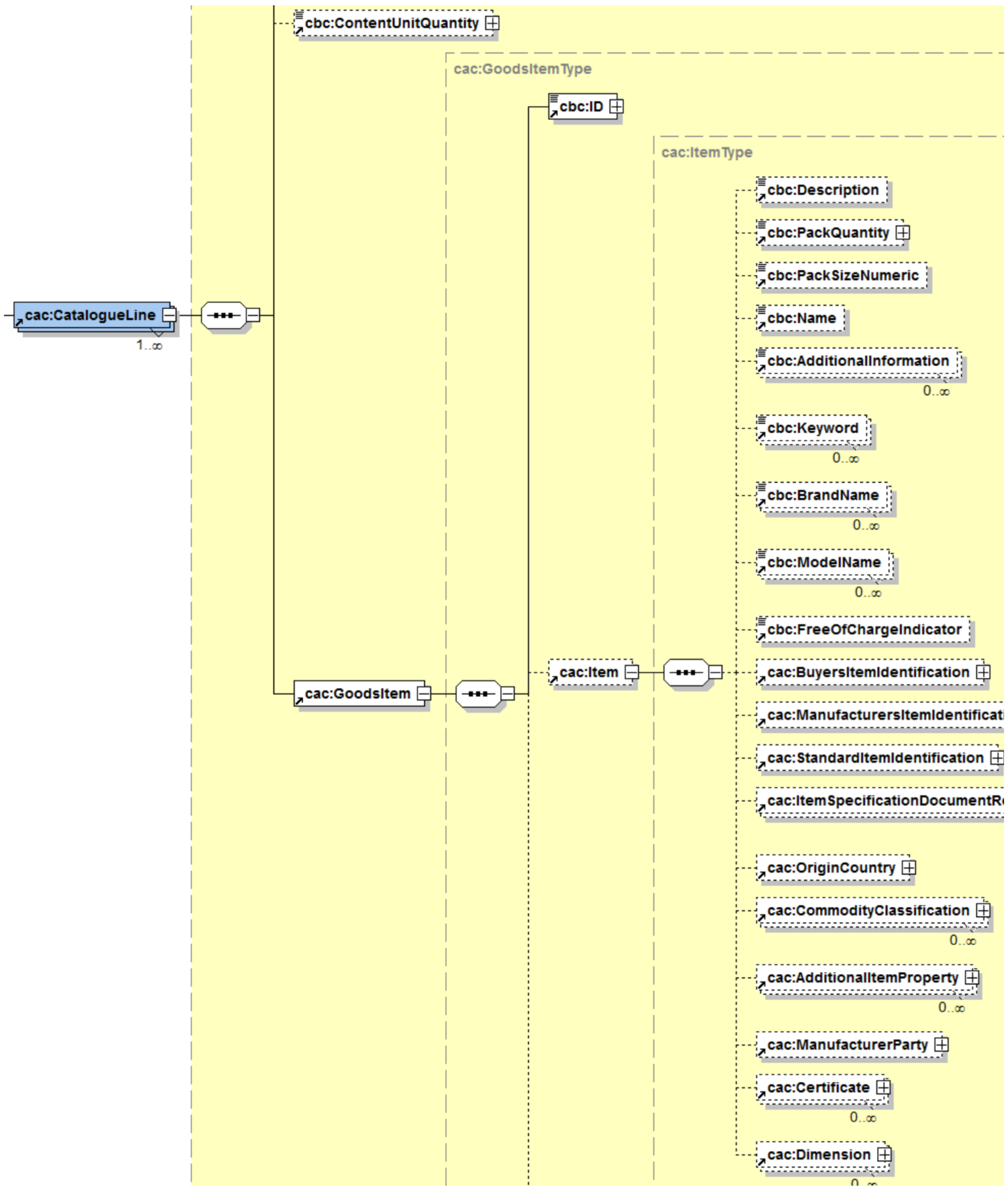


Figure 5 UBL Graphical representation for Catalogue Line schema

UBL defines the *Catalogue* document schema such that it would contain a set of *Catalogue Lines* corresponding to products traded by a party. Figure 5 shows a graphical representation of a portion of the *Catalogue Line* data schema. The elements referred with a “cbc” prefix are BBIEs defined in the common BBIE library provided by UBL. Likewise, the ones referred with a “cac” prefix are ABIEs from the common ABIE library. Although we preserved the original schema provided by UBL, we did some modifications based on the requirements of our use cases. First, in order to start with a relatively simple data model, we removed unnecessary elements that are not required in our use cases, mainly for the Micuna use case. Nevertheless, we did some modifications as well. In Annex B, we initially had indicated the Micuna use case data elements that do not have a corresponding in UBL with a *TBD* mark. Mainly, those are the elements for which we customize the original UBL data model. For example, we added concepts related to payment and delivery conditions to the company concept. The current version of the data model for the Catalogue Registry can be found at [8].

2.2.3 Ontology Schema

This sub-section provides a high-level visualization of the ontology schema. It follows the key concepts listed in sub-section 2.2.1 *Basic Elements* and definitions in sub-section 2.2.2 *Universal Business Language*. A high-level visualization of the catalogue domain ontology is illustrated in Figure 6. For visibility reasons, only the most important concepts are shown in the figure, and the object properties as well as data type properties are not detailed. More detailed information can be found in the separate deliverable D2.2 Semantic Modelling of Manufacturing Collaboration Assets.

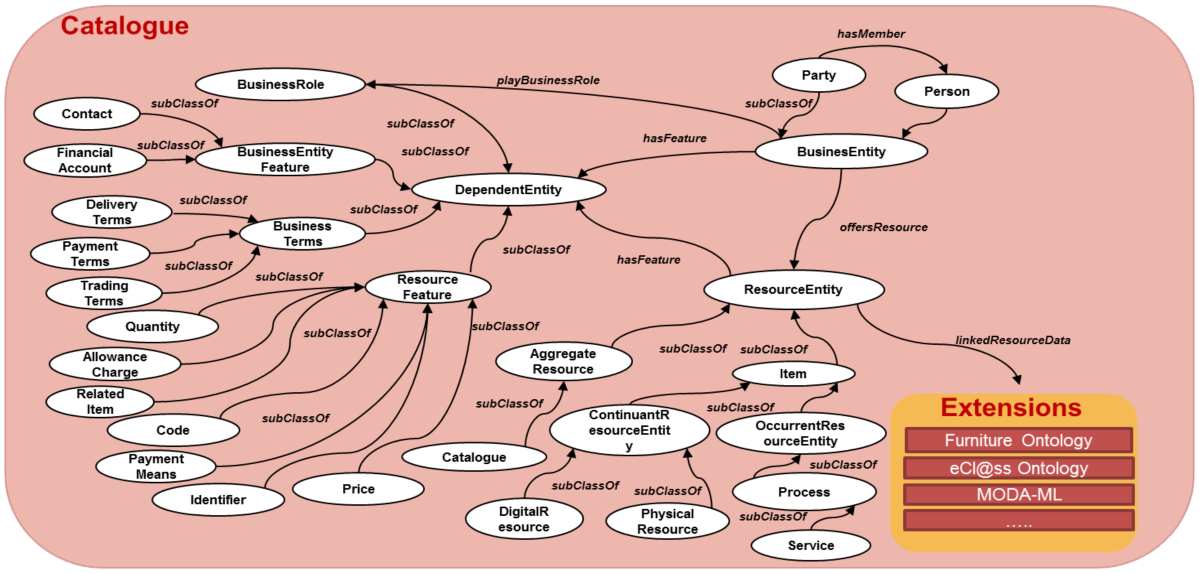


Figure 6 High-Level View of the Catalogue ontology schema

[8] https://github.com/nimble-platform/catalog-service-srdc/tree/master/business/ubl-data-model/src/main/schema/NIMBLE-UBL-2.1-Catalog-Subset

As depicted in the figure above, the key concepts described in sub-section 2.2.1 are included. In a high-level view, a Business Entity can offer some Resource Entities. The Business Entity as well as Resource Entities can have properties which are detailed in Dependent Entities. In addition, the resource entity can be extended with other resources such as eClass ontology, Furniture taxonomy in the furniture domain or MODA-ML in the textile domain for detailed domain specific resource specification. The extension is applicable mainly thanks to the linked data mechanisms.

Most of the concepts in the ontology schema are derived from the UBL standard data model, which is specified in sub-section 2.2.2. The catalogue related concepts and properties in the UBL data model are tailored and transformed into ontology classes and object/data type properties in the above shown catalogue ontology schema. For example, the UBL concepts of *Catalogue* or *Item* are transformed into the ontology classes *Catalogue* and *Item*. In order to describe commercial conditions for the resource entity, the payment, price, warranty as well as delivery related concepts from UBL such as *PaymentTerms* are included. Many other concepts, which are derived from the UBL data models, such as *Period*, *Address*, are for the reason of visibility not listed in the above diagram. The relationships that are in the form of ontology object properties and datatype properties are for the same reason not detailed in the above diagram. For example, a *Catalogue* can have many *Catalogue Lines*. Each *Catalogue Line* can have its own *Warranty Information*, *Warranty Party* and *Warranty Validity Period*. In the example, the Warranty Information is a datatype property in the ontology, the *Warranty Party* and *Warranty Validity Period* are both object properties with links to the respective ontology class *Party* or *Period*.

After the catalogue related concepts are derived from the standard UBL data model, these concepts are re-arranged based on the top-level ontology BFO as well as the key concept categories described in the sub-section 2.2.1.

2.3 Linking the Taxonomies with the Data Model

The UBL data model, specifically the *ItemType* concept, which is used to annotate a product, has an appropriate sub-concept to link the product with an external category. This sub-concept is a common ABIE called *CommodityClassificationType*. It further contains additional BBIEs to provide the coded value of the category along with a URI reference to the category definition in the target taxonomy.

Once an *ItemType* is associated with a category, the properties defined by the category are added to the *ItemType* as *ItemProperties*, which is another common ABIE that could be used to express diverse types of primitive values such as text, number or binary objects. This association is either realized on the UI when users publish products one by one or once a catalogue template is uploaded to NIMBLE. Ingestion modalities are presented in the next section in detail. Figure 7 shows how the knowledge obtained from the taxonomies are integrated into the UBL data model.

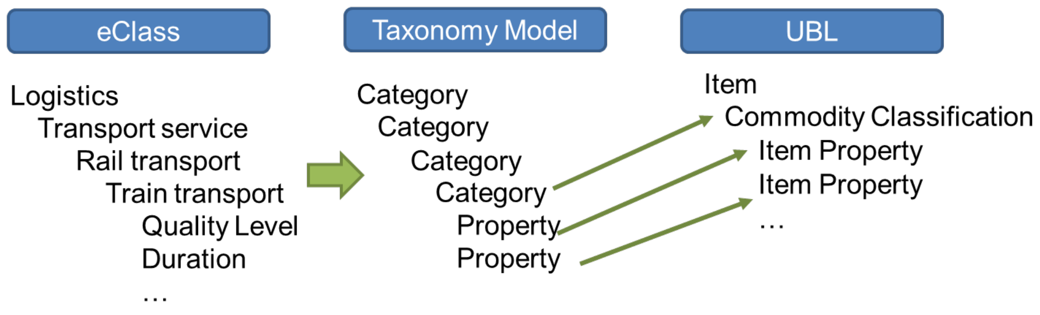


Figure 7 Integration of eClass resources into the UBL data model

3 Ingestion Design

We follow the approach introduced by the ebXML Registry standard for managing the information about product data. An ebXML Registry manages any content type and the standardized metadata that describes it [3]. While the former concept is named as *RegistryItem*, the latter one is called *RegistryObject.* While *RegistryItems* might differ in the type of content such as image, video, XML document, etc., *RegistryObjects* have a standard representation. In order to be able to enable operations that would work on heterogeneous *RegistryItems,* ebXML introduces two types of storage concepts dealing with these two types of data: *repository* and *registry*.

Similarly, NIMBLE needs to work with product definitions that have heterogeneous formats and structures. Thus, the Catalogue Registry manages two types of information about products:

1. Product data: Such data would contain any information about the product including any type of binary attachments. Structure of the product data might differ from sector to sector or company to company. Even, there might not be any structured representation of products.

2. Metadata: Metadata are extracted from the product data and stored as key-value pairs.

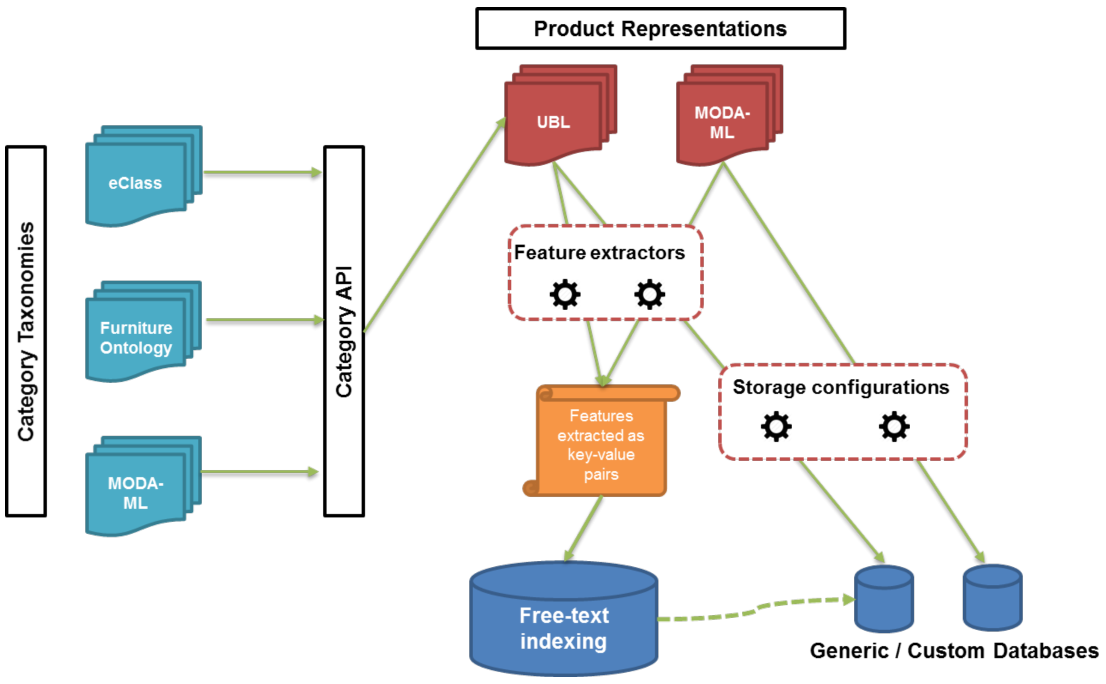


Figure 8 Storing product data and metadata

Figure 8 depicts the process for storing product data and metadata. The top-level elements indicate that the initial product data might have different structures, that are compliant with different standards or that are proprietary. NIMBLE allows providers to use custom *Feature Extractors* to extract metadata from product data with different structures. For the time being, while the product data might reside in different repositories, extracted properties are kept in a single repository to be able to provide a unified search over the stored products. However, in the future, we might adopt an approach where the metadata is also stored in a distributed way considering the requirements of NIMBLE federations. See the federation-related future work in Section 6.

As depicted in Figure 7, categories are represented in the taxonomy model before being integrated into the product data model. For the time being, we have wrapped eClass with the taxonomy model and as future work we will develop wrappers for the sector-specific taxonomies.

3.1 Persistence Modalities

We have two means of storage as an implementation of the design we just introduced. The first is for the data and the second for the metadata extracted from product information. Once a product is published to NIMBLE, first the original content is stored in a database with all associated information. Then, the feature extractors run on the published data and extract metadata as key-value pairs from heterogeneous product data.

Metadata are used to enable the discovery process of products on NIMBLE. After the initial discovery, the details about the product can be retrieved from the relevant repository. Nevertheless, NIMBLE also offers alternative search capabilities as will be described in *D3.3 - Product and Service Search Engine and Search Mediator*.

Having UBL as the common data model for representation of product data, the default repository of NIMBLE is a UBL-compliant relational database keeping the product data in a structured way.

On the other hand, we use Apache Marmotta[9] to store both the actual content and metadata of products. Marmotta uses a triple store to persist its data. The Triple store manages the data as RDF triples [4]. To be able to transform UBL-compliant product data into RDF, we used a tool called Ontmalizer[10]. Ontmalizer is able to transform any XML document to RDF as long as the XML document is compliant with an XSD schema, which is the case for UBL-based product data. Once the data is in RDF format, it is sent to Marmotta to be stored.

In addition to providing a triple store, Marmotta has an add-on module using Apache Solr[11] (a.k.a Solr) for text-based indexing of the data. A Solr index includes a set of index field definitions each of which corresponds to a different metadata about the submitted document. So, the way to index a document to a Solr index is to extract information for each index field from the original document and put the extracted information into relevant index field along with a reference to the original document. It is also possible to define a specific index field containing the entire text of the document.

Marmotta supports a query language, called LDPath[12], to query the RDF data stored in the triple store and populate a Solr index with results of the query. An LDPath expression contains a set of (relative) RDF paths that correspond to an index field. So, this means that LDPath is the default *Feature Extractor* for the UBL-based product data.

```
@prefix cat: <urn:oasis:names:specification:ubl:schema:xsd:Catalogue-
2#> ;

@prefix cac:
<urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponent
s-2#> ;

@prefix cbc:
<urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-
2#> ;

@filter rdf:type is cac:ItemType ;
```

[9] http://marmotta.apache.org/

[10] https://github.com/srdc/ontmalizer

[11] http://lucene.apache.org/solr/

[12] http://marmotta.apache.org/ldpath/language.html

```
item_name = cbc:Name :: xsd:string ;

item_description = cbc:Description :: xsd:string ;

item_complete_images = cbc:ItemConfigurationImages :: xsd:string;

df_d = cac:AdditionalItemProperty[cbc:ValueQualifier is
"REAL_MEASURE"] / fn:dynamic(cbc:Name, cbc:Value) :: lmf:dynamic
(dynamicField="*_d", solrType="double") ;

df_s = cac:AdditionalItemProperty[cbc:ValueQualifier is "STRING"] /
fn:dynamic(cbc:Name, cbc:Value) :: lmf:dynamic (dynamicField="*_s",
solrType="string") ;

df_st = cac:AdditionalItemProperty[cbc:ValueQualifier is
"STRING_TRANSLATABLE"] / fn:dynamic(cbc:Name, cbc:Value) ::
lmf:dynamic (dynamicField="*_st", solrType="string") ;
```

Listing 1 LDPath instance for extracting properties from UBL-based product data

Listing 1 is the LDPath instance being used to extract properties from UBL-based documents. The instance contains a *@filter* command to filter the RDF triples with type *cac:ItemType*. This class is the ontology class used to annotate a resource as a product. The resources filtered by this command are the root resources of which details are obtained via the RDF path definitions following the filter command. Each RDF path definition specifies the name of the index field and the assignment values identified by following the subsequent (relative) path statements starting from the root resource. The expected data type of the values finalizes each statement. For example the statement *item_name = cbc:Name :: xsd:string*; (see Listing 1) will return the result “Super Bathroom” on RDF data given in Listing 2.

```
<rdf:Description
rdf:about="urn:oasis:names:specification:ubl:schema:xsd:Catalogue-
2#INS9709254_ItemType_1">

      <rdf:type
rdf:resource="urn:oasis:names:specification:ubl:schema:xsd:CommonAggre
gateComponents-2#ItemType"/>

      <ns3:Name>Super Bathroom</ns3:Name>

      <ns2:AdditionalItemProperty
rdf:resource="urn:oasis:names:specification:ubl:schema:xsd:Catalogue-
2#INS9709254_ItemPropertyType_1"/>

</rdf:Description>

<rdf:Description
rdf:about="urn:oasis:names:specification:ubl:schema:xsd:Catalogue-
2#INS9709254_ItemPropertyType_1">

      <rdf:type
rdf:resource="urn:oasis:names:specification:ubl:schema:xsd:CommonAggre
gateComponents-2#ItemPropertyType"/>

      <ns3:Name>Wall Tile</ns3:Name>

      <ns3:ValueQualifier>Text</ns3:ValueQualifier>

      <ns3:Value>Green tiles</ns3:Value>

</rdf:Description>
```

Listing 2 An example product data in RDF format

Normally, the index fields for a Solr index are defined at the beginning and the documents to be indexed are processed for those fields. However, in NIMBLE products would have distinctive properties to be represented as index fields that could be retrieved either from product property taxonomies or specified by the users themselves. So, it is not feasible to create an index field for each distinct property beforehand. Therefore, we updated the initial LDPath implementation with new parsing functions to be able to create dynamic index fields, which is supported by Solr. For example, the *cac:AdditionalItemProperty[cbc:ValueQualifier is "STRING"] / fn:dynamic(cbc:Name, cbc:Value) :: lmf:dynamic (dynamicField="*_s", solrType="string")* command in the example LDPath instance, creates a dynamic index field of type *string* if the *value qualifier* of an *additional item property* is *STRING.* Figure 9 depicts the ingestion flow elaborated in this section.

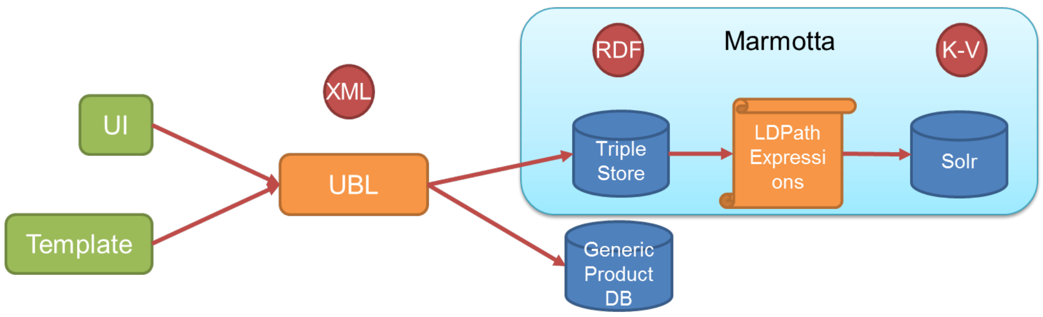


Figure 9 Ingestion flow for product data and metadata

By making use of Marmotta-Solr collaboratively we are able to store semantic metadata of products in a free-text engine that combines the faceted search paradigm with the semantic search paradigm.

3.2 Ingestion Modalities

As the Catalogue Registry has similar capabilities with available eCommerce platforms, we investigated how the existing platforms enable their users to publish product information. Specifically, we investigated both international platforms (Alibaba and Amazon) as well as a local one (Hepsiburada, a Turkish one). The common process among all these platforms is as follows: Ask users to:

1. Choose a product category first
2. Download a spreadsheet based template containing product properties for the selected category
3. Fill in the template for distinct products and provide alternative values for particular properties e.g. different colours alternatives for a tile represented as a single product
4. Upload the template

This approach enables users to publish products in a bulk manner. On the other hand, some of them allow product publishing one by one but still after asking users to select a category. Then users specify values for pre-defined properties proposed for the selected category. In either case, users are able to specify custom properties.

From the user experience point of view, new applications / platforms are recommended to provide similar ways of doing things with the existing applications / platforms having similar aims. Because, users develop usage patterns on such platforms and they would like to use the same pattern instead of learning something from scratch. Therefore, we also make use of the aforementioned ingestion modalities.

4 API Design

In this section, we provide details about the current API of the Catalogue Registry. There are two main APIs for this component namely, a REST API and a Java API. The REST API enables the consumption of the Catalogue Registry capabilities by executing HTTP-based queries. The REST API delegates the task to the Java API to realize the job expected from the service. In this deliverable, we report the current status of the API. The latest status update will always be on GitHub [13].

4.1 REST API

Within the Catalogue Registry itself, there are two sets of REST services for the time being: *ProductCategoryController* and *Catalogue Controller.*

ProductCategoryController enables the retrieval of different kinds of information about the product category taxonomies.

Table 3 ProductCategoryController services

Service Name	Service Path	Service Description
getCategoryById	/catalogue/category/{categoryId} PathParam: categoryId	Returns a category class with all details
getDefaultCatalogue	/catalogue/{partyId}/default PathParam: partyId	Returns the default catalogue for the given party
getCategoriesByName	/catalogue/category	Returns a list of categories for a given

[13] https://github.com/nimble-platform/catalog-service-srdc

	QueryParam: categoryName	keyword
getSubCategories	/catalogue/category/{parentCategoryId} /subcategories PathParam: parentCategoryId	Returns the child category classes for the specified parent class

Although the current set of services included in *ProductCategoryController* works only on the built-in eClass-based taxonomy, we will soon add support for multiple taxonomies

CatalogueController includes services for management of products through CRUD (Create-Read-Update-Delete) functionalities.

Table 4 CatalogueController services

Service Name	Service Path	Service Description
getCatalogueByUUID	/catalogue/{uuid} PathParam: uuid	Returns the catalogue identified by the given identifier
addCatalogue	/catalogue RequestBody: catalogueJson	Stores the given catalogue
updateCatalogue	/catalogue RequestBody: catalogueJson	Updates a catalogue with the new data
deteleCatalgoue	/catalogue/{uuid}	Deletes the catalogue with the given identifier
downloadTemplate	/catalogue/template RequestParam: categoryId	Generates a template for the product category specified by the given identifier
uploadTemplate	/catalogue/template/upload RequestParam: file	Uploads the template

For the time being, *CatalogueController* provides CRUD capabilities at the catalogue-level. In the future, we will include CRUD capabilities also at the product level so that users will be able to manage individual products.

In addition to REST services developed in the scope of Catalogue Registry implementation, we are also able to make use of the REST services provided by Marmotta and Solr. Marmotta provides CRUD services for RDF-based data management. In addition to the CRUD service on the Solr-specific documents, Solr also provides advanced search capabilities. These external services, however, are mainly used by the internal NIMBLE implementation.

4.2 Java API

Similar to the REST API, the current Java API has two main modules namely *CatalogueService* and *ProductCategoryService*.

ProductCategoryService includes methods with one-to-one mapping to the services included in *ProductCategoryController*. As a future work, we will improve the category API for extension of the base taxonomy through crowd-sourcing. In this way, on one hand, users will be able recommend new categories and new properties for existing categories. On the other hand, authorized parties will be able to check the requests of users and once the request is approved the recommendation will be integrated into the NIMBLE taxonomy permanently.

CatalogueService has more capabilities than its REST counterpart, i.e *CatalogueController,* as it includes CRUD methods for both UBL-based catalogues and other custom formats.

Table 5 CatalogueService methods

Method Name	Method Parameters	Method Description
addCatalogue	*CatalogueType catalogue*	Stores the given catalogue both to the default repository, triple store; extracts the metadata and store it in the dedicated Solr index
addCatalogue	String catalogueXML	Stores the given catalogue in UBL-compliantXML format both to the default repository, triple store; extracts the metadata and store it in the dedicated Solr index
getCatalogue	String uuid	Returns the catalogue identified by the given identifier
getCatalogue	String id String partyId	Return the catalogue specified by *id* for the party specified by *partyId*
updateCatalogue	*CatalogueType* catalogue	Updates a catalogue with the new data
deleteCatalogue	String uuid	Deletes the catalogue with the given identifier
addCatalogue	String catalogueXML Standard standard	Adds the catalogue in XML format compliant with the specified standard into the relevant repository

addCatalogue	T catalogue Standard standard	Adds the catalogue compliant with the given standard into the relevant repository
getCatalgoue	String uuid Standard standard	Returns the catalogue compliant with the specified standard and with the given identifier
getCatalogue	String id String partyId Standard standard	Returns the catalogue compliant with the specified standard and with the given identifier for the specified party
deleteCatalogue	String uuid Standard standard	Deletes the catalogue compliant with the given standard for the given identifier

5 User Interfaces

5.1 Mock-ups

For the user interfaces, we followed an iterative approach by first presenting mock-ups to end users. We prepared interactive mock-ups with the Invision tool[14]. In addition to interaction, users were also able to comment on specific parts of the mock-up indicating their feedback. Annex C includes the mock-ups for the Catalogue Registry. The complete mock-up package is available online at [15].

5.2 User Interfaces

After the initial mock-up phase, we developed the initial UI of the Catalogue Registry. For the time being, we developed views for product publishing. As future work, we will develop views also for management of the products after the initial publishing. Annex D shows mock-ups for category search and product publishing.

6 Future Work

As future work, we will continue elaborating the API in the scope of T2.3 and work on the implementation of the API as stated in Section 4 and user interfaces in Section 5. We will adopt

[14] https://www.invisionapp.com/

[15] https://invis.io/7W9UWPTV4

Swagger[16] for improved documentation and testing of the REST API. Although a central management of *Feature Extractors* and *Storage Configurations* is needed, Figure 10 shows the implementation status of various modules such that while green arrows indicate that the connection between the components is already established e.g. eClass is wrapped with the taxonomy model and integrated into the UBL product representations; a wrapper is still needed for the furniture ontology.

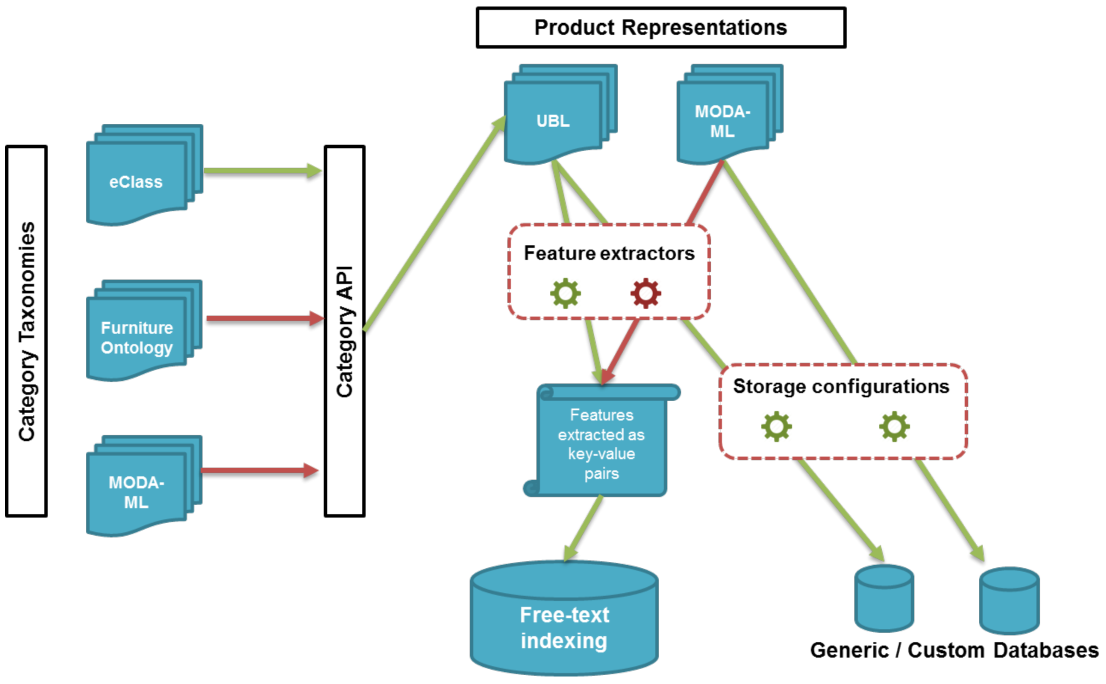


Figure 10 Implementation status of feature extractors and database setups

We will also work on federation-related improvements of the Catalogue Registry. We will identify the federation-related requirements and corresponding system design. We will decide the way a federation will be formed; in which ways federation members will be aware of each other; and how the federation members will be synchronized if at all.

Lastly, we will improve the Catalogue Registry with the security enablers to be developed in WP6. Enablers such as REST service security, resource access control mechanisms and automatic authentication will be integrated.

7 Summary

In this deliverable, we report on the design decisions about the catalogue ingestion mechanism in NIMBLE. We elaborate on the mechanism for annotating the products with semantically relevant metadata. In this respect, we mention product category taxonomies as

[16] http://swagger.io/

collections of product categories associated with a set of category-specific properties. After presenting the data representation, we show how we utilize these taxonomies and link them with the catalogue data model.

From an architectural point of view, we make a distinction between data and metadata; and provide details about the ways and means to deal with those two kinds of information. Lastly, we provide details about the latest version of the Catalogue Registry API along with some screenshots of the user interfaces.

8 References

[1] Stolz, A., Rodriguez-Castro, B., Radinger, A., & Hepp, M. (2014, May). PCS2OWL: A generic approach for deriving web ontologies from product classification systems. In European Semantic Web Conference (pp. 644-658). Springer International Publishing.

[2] Smith, B., Almeida, M., et al. (2015)Basic Formal Ontology 2.0 SPECIFICATION AND USER'S GUIDE. http://purl.obolibrary.org/obo/bfo/Reference. Last visited 2017-06-30.

[3] OASIS ebXML RegRep Version 4.0 Part 0: Overview Document. 25 January 2012. OASIS Standard. Retrieved from http://docs.oasis-open.org/regrep/regrep-core/v4.0/os/regrep-core-overview-v4.0-os.html, 26 June 2017.

[4] Klyne, Graham, Jeremy J. Carrol, and Brian McBride. "RDF 1.1 Concepts and Abstract Syntax." RDF 1.1 Concepts and Abstract Syntax. 25 Feb. 2014. Retrieved from https://www.w3.org/TR/rdf11-concepts/, 28 June 2017.

9 Appendices

9.1 Annex A - Excerpt of Entities Involved in the MICUNA Use Case

The table below includes some relevant terms used in the scenarios of the MICUNA use case. Many of these terms will be common to all use cases. They have been added from domains that can be differentiated: (i) products, (ii) components/materials, (iii) production operations/phases and (iv) trade concepts.

X: means the concept is registered in that taxonomy with the same meaning.

(X): means the concept is in the taxonomy but there are only specific instances related to the term instead of a general class representing it, or the meaning is not exactly the same as the one in the MICUNA domain

-: means the concept is missing or it has a different meaning that the one in the MICUNA domain

It should be noted that the considered taxonomies are quite different in terms of levels while here the only aspect considered is the presence or not of an example collection of terms.

	eClassOWL 5.1.4	CPV 2008	ETIM 6.0	Google Prod Tax
adhesive	X	X	X	X
assembly	X	X	X	-
balance	X	X	X	(X)
board	(X)	X	-	(X)
business	X	-	-	-
carrier	(X)	-	(X)	-
catalogue	X	(X)	X	-
certification	X	(X)	-	-
composition	X	-	-	-
cradle	(X)	X	-	X
crystal	X	X	X	X
delay	X	-	(X)	-
delivery	X	(X)	X	-
din	X	-	-	-
discount	-	-	-	-
donation	(X)	-	-	-
environment	(X)	(X)	-	-
finishing	X	X	-	X
fitting	X	-	(X)	(X)
flexible	X	-	X	-
guarantee	X	-	-	-
handcrafted	-	(X)	-	(X)
installation	X	X	X	-
insurance	X	(X)	-	X
iso	(X)	-	(X)	-
labelling	(X)	(X)	(X)	(X)
law	X	-	-	-
machining	X	(X)	-	-
negotiation	X	-	-	-
norm	X	X	-	-
order	X	-	-	-
outsourcing	(X)	-	-	-
packaging	X	(X)	X	-
paint	X	X	(X)	X
payment	X	-	-	-
piece	X	(X)	(X)	(X)
plastic	X	X	X	-
polishing	X	-	(X)	-
price	X	-	X	-

quality	X	X	(X)	-
quantity	X	X	X	-
regulation	X	-	-	-
renovation	X	X	-	-
repayment	-	-	-	-
safety	(X)	(X)	X	(X)
sanding	X	(X)	-	(X)
sensor	X	X	X	X
shapping	-	-		-
stock	X	-	(X)	-
storage	X	(X)	X	(X)
supplier	X	(X)	-	-
technology	X	X	(X)	-
textile	X	X	X	(X)
tool	X	X	X	(X)
transport	X	X	(X)	-
unit	X	-	X	-
varnishing	X	-	-	X
warehouse	X	(X)	(X)	(X)
wheel	X	X	X	X
wood	X	X	(X)	X

9.2 Annex B – Initial Alignment of the Data Elements of the Micuna Use Case with the UBL Elements

PRODUCT SUPPLIER DATASHEET

GENERAL INFORMATION		
Name		*cac:Party/cac:PartyName/cbc:Name*
Code		*cac:Party/cbc:IndustryClassificationCode*
NIF/VAT		*cac:Party/cac:PartyTaxScheme/cac:TaxScheme*
Address		*cac:Party/cac:PostalAddress*
Zip code		*cac:Party/cac:PostalAddress/cbc:PostalZone*
Town		*cac:Party/cac:PostalAddress/cbc:CityName*
Region		*cac:Party/cac:PostalAddress/cbc:Region*
Contact person		*cac:Party/cac:Person*
Position of contact person		*cac:Party/cac:Person/cbc:JobTitle*
Email		*cac:Party/cac:Contact/cbc:ElectronicMail*
Telephone		*cac:Party/cac:Contact/cbc:Telephone*
Fax		*cac:Party/cac/Contacy/cbc:Telefax*
Classification		*cac:Party/cbc:IndustryClassificationCode*
Supplier facilities		*cac:Party/cac:PhysicalLocation*
ISO compliance		*cac:Party/cac:PartyLegalEntity*

OSHAS certification		*cac:Party/cac:PartyLegalEntity*
EFQM implementation		*cac:Party/cac:PartyLegalEntity*
Awards received		*TBD*
Penalty clauses for late delivery		*cac:DeliveryTerms/cbc:Amount*
Quality indicators		*TBD*
Technical advice		*cac:ItemSpecificationDocumentReference*
TRADE CONDITIONS		
Method of payment		*cac:PaymentMeans*
Payment conditions		*cac:PaymentTerms*
Payment days		*cac:PaymentTerms/cac:SettlementPeriod/cbc:DurationMeasure*
Bank entity		*cac:FinancialAccount/cac:FinancialInstitituonBranch/cac:FinancialInstitution*
Bank account number		*cac:FinancialAccount*
Discounts		*cac:AllowanceCharge/cbc:Amout*
	Trade discounts	*cac:AllowanceCharge/cbc:AllowanceChargeReasonCode*
	Rappel (variable)	*cac:AllowanceCharge/cbc:AllowanceChargeReasonCode*
	Cash discount	*cac:AllowanceCharge/cbc:AllowanceChargeReasonCode*
Delivery conditions		
	General comments	*cac:DeliveryTerms/cbc:SpecialTerms*
	Delivery period	*cac:Delivery/cac:PromisedDeliveryPeriod*
	Freight type	*cac:Shipment/cac:ShipmentStage/cbc:TransportModeCode*
	Insurance	*cac:Delivery/cbc:InsuranceValueAmount*
	Services	*cac:Shipment/cbc:HandlingInstructions or cbc:DeliveryInstructions*
	Other fees	*cac:Shipment/cac:DeclaredForCarriageValueAmount*

LOGISTICS SUPPLIER DATASHEET

GENERAL INFORMATION		
Name		*Same as above*
Code		*Same as above*
NIF/VAT		*Same as above*
Address		*Same as above*
Zip code		*Same as above*
Town		*Same as above*
Region		*Same as above*
Contact person		*Same as above*

Position of contact person		*Same as above*
Email		*Same as above*
Telephone		*Same as above*
Fax		*Same as above*
Classification		*Same as above*
Supplier facilities		*Same as above*
ISO compliance		*Same as above*
Safety stock		*cac:ItemManagementProfile/cbc:MinimumInventoryQuantity*
Penalty clauses for late delivery		*Same as above*
Quality indicators		*Same as above*
Technical advice		*Same as above*
TRADE CONDITIONS		
Method of payment		*Same as above*
Payment conditions		*Same as above*
Payment days		*Same as above*
Bank entity		*Same as above*
Bank account number		*Same as above*
Discounts		*Same as above*
	Trade discounts	*Same as above*
	Rappel	*Same as above*
	Cash discount	*Same as above*
Service conditions		
	General comments	*Same as above*
	Cargo days	*cac:Delivery/cac:PromisedDeliveryPeriod/cbc:DurationMeasure*
	Freight type	*Same as above*
	Insurance	*Same as above*
	Services	*Same as above*
	Other fees	*Same as above*
SERVICE SPECIFICATIONS		
Delivery method		*cac:Shipment/cac:ShipmentStage/cbc:TransportModeCode*
Destination		*cac:DeliveryCustomerParty*
Insurance		*cac:Shipment/cbc:InsuranceValueAmount*
Rate		*cac:Shipment/cac:DeclaredForCarriageValueAmount*
Quality indicator		*TBD*
Certifications		*cac:Item/cac:Certificate*
General features		*cac:Shipment/cbc:Information*

PRODUCT SPECIFICATIONS

Name	*cac:Item/cbc:Name*

Code	*cac:Item/cac:CommodityClassification*
Price	*cac:Price/cbc:PriceAmount*
Product features	*cac:Item/cac:AdditionalItemProperty*
Minimum order quantity	*cac:CatalogueLine/cbc:MinimumOrderQuantity*
Materials	*cac:Item/cac:ItemSpecificationDocumentReference*
Certifications	*cac:Item/cac:Certificate*
Technical safety datasheet	*cac:Item/cac:ItemSpecificationDocumentReference*
Tolerance/Allowed measurement deviations	*cac:Item/cac:Dimension/cbc:MinimumMeasure, cbc:MaximumMeasure*
Packaging type	*cac:GoodsItem/cac:ContainingPackage/cbc:PackagingTypeCode*
Product image	*cac:Item/cac:ItemSpecificationDocumentReference*
Usual delivery period	*cac:Delivery/cac:EstimatedDeliveryPeriod*
Delivery period in non-conformity claims	*TBD*
Graduated prices by quantity	*TBD*
First product sample free of charge	*cac:InvoiceLine/cbc:FreeOfChargeIndicator*
Quantity per package	*cac:GoodsItem/cac:ContainingPackage/cbc:Quantity*
Technical product datasheet	*cac:Item/cac:ItemSpecificationDocumentReference*
Product guarantee	cac:CatalogueLine/cbc:WarrantyInformation
Safety stock	*cac:ItemManagementProfile/cbc:MinimumInventoryQuantity*

9.3 Annex C – Mock-ups for Catalogue Registry

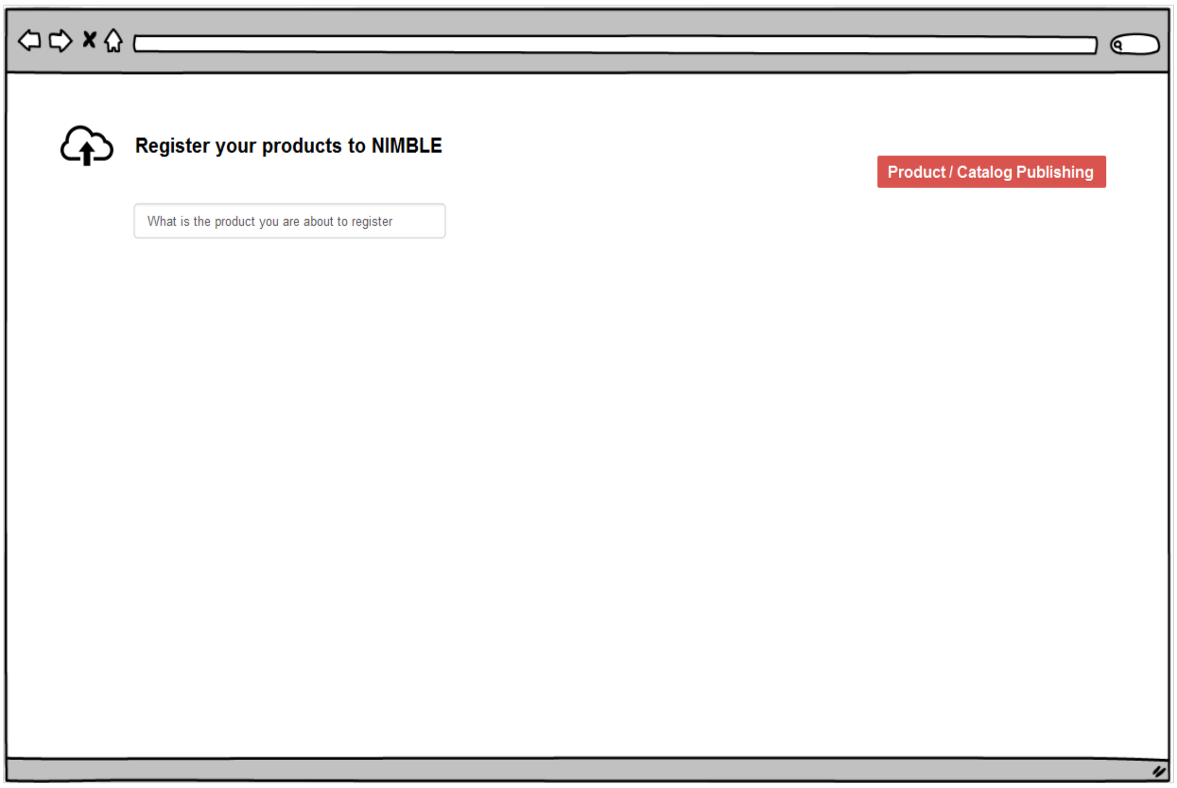

Register your products to NIMBLE
What is the product you are about to register
Product / Catalog Publishing

Figure 11 Search for product categories with a keyword

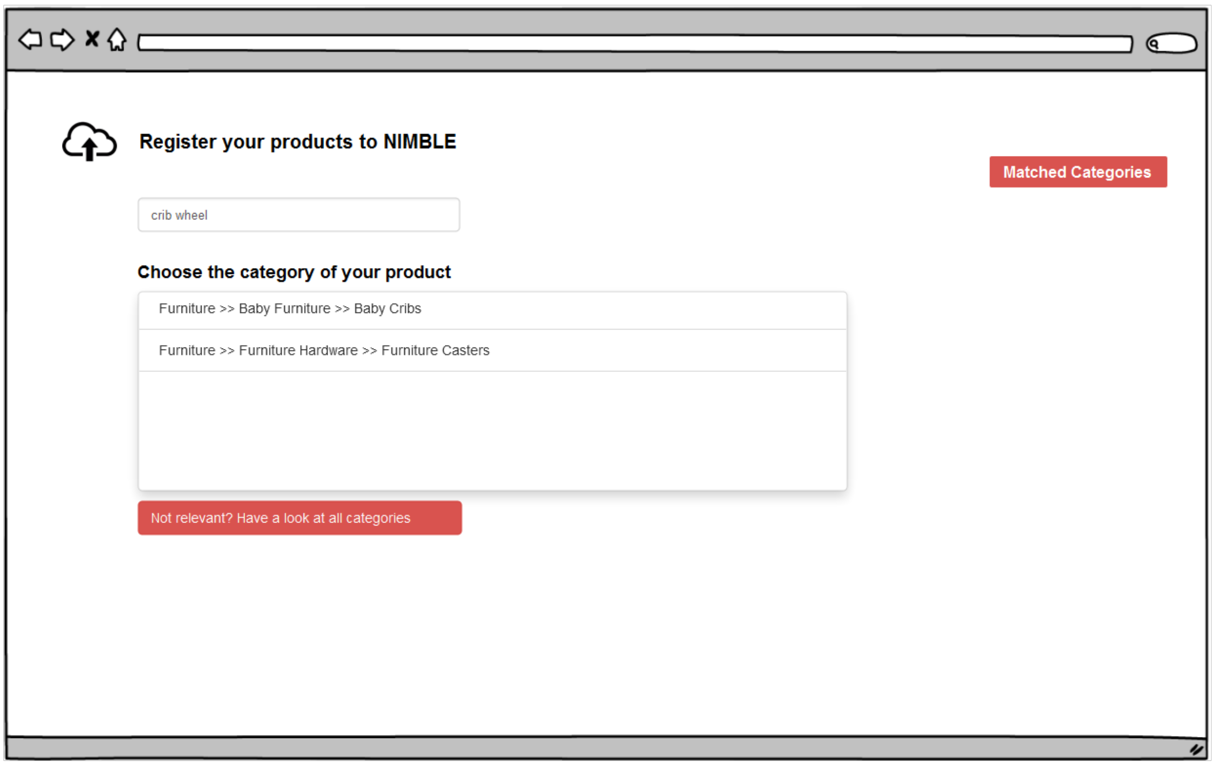

Register your products to NIMBLE
Matched Categories
crib wheel
Choose the category of your product
Furniture >> Baby Furniture >> Baby Cribs
Furniture >> Furniture Hardware >> Furniture Casters
Not relevant? Have a look at all categories

Figure 12 Categories listed for the provided keyword

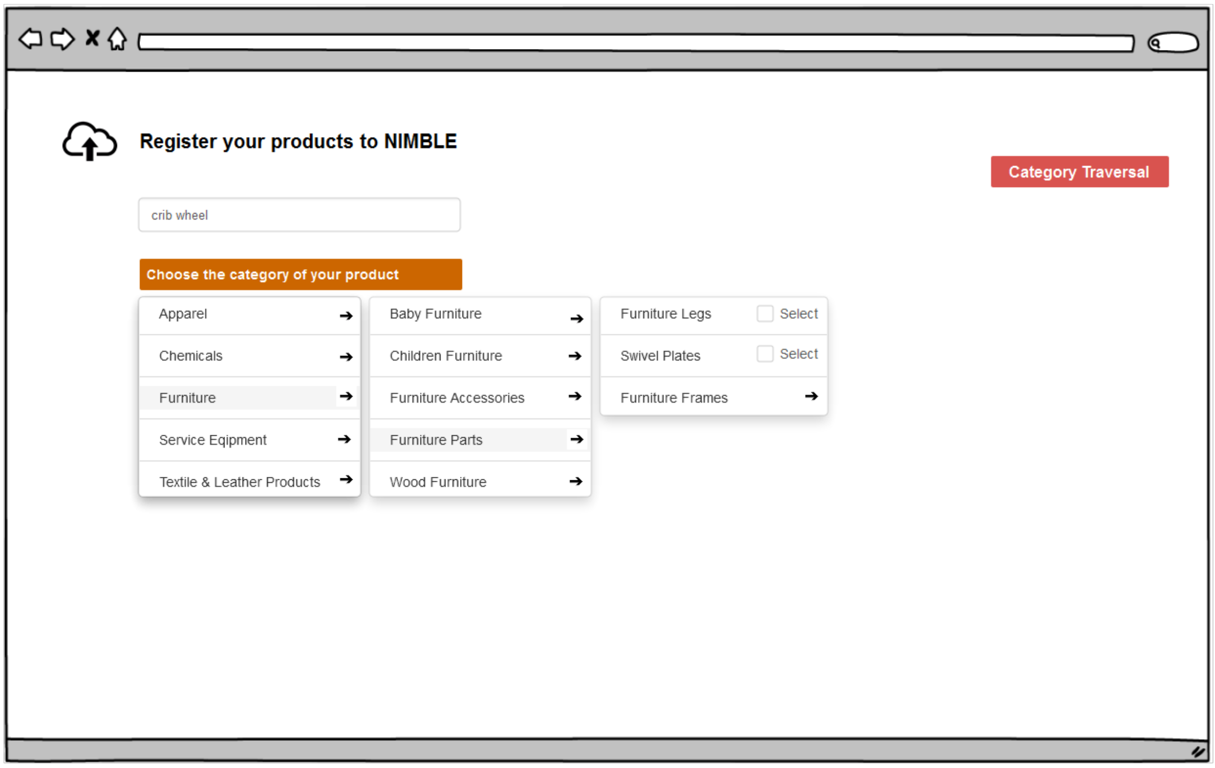
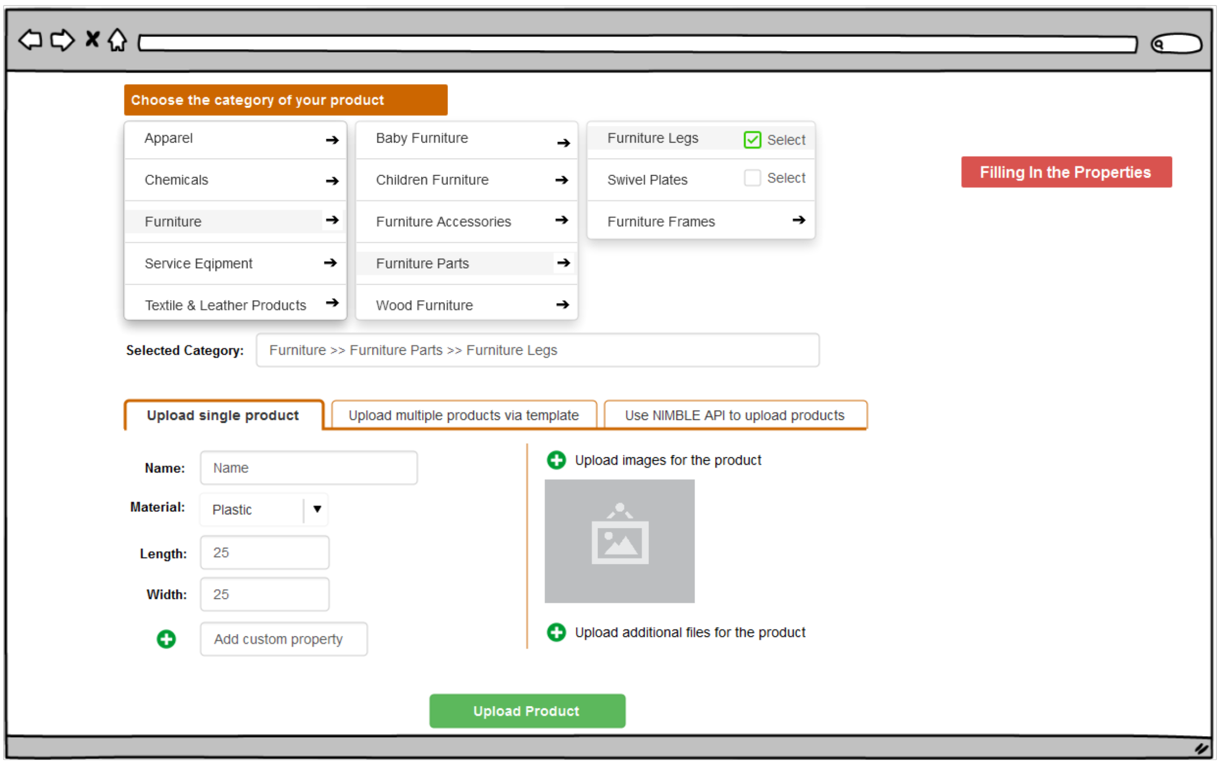


Figure 13 Category selection by traversing the taxonomy hierarchy



Selected Category: Furniture >> Furniture Parts >> Furniture Legs

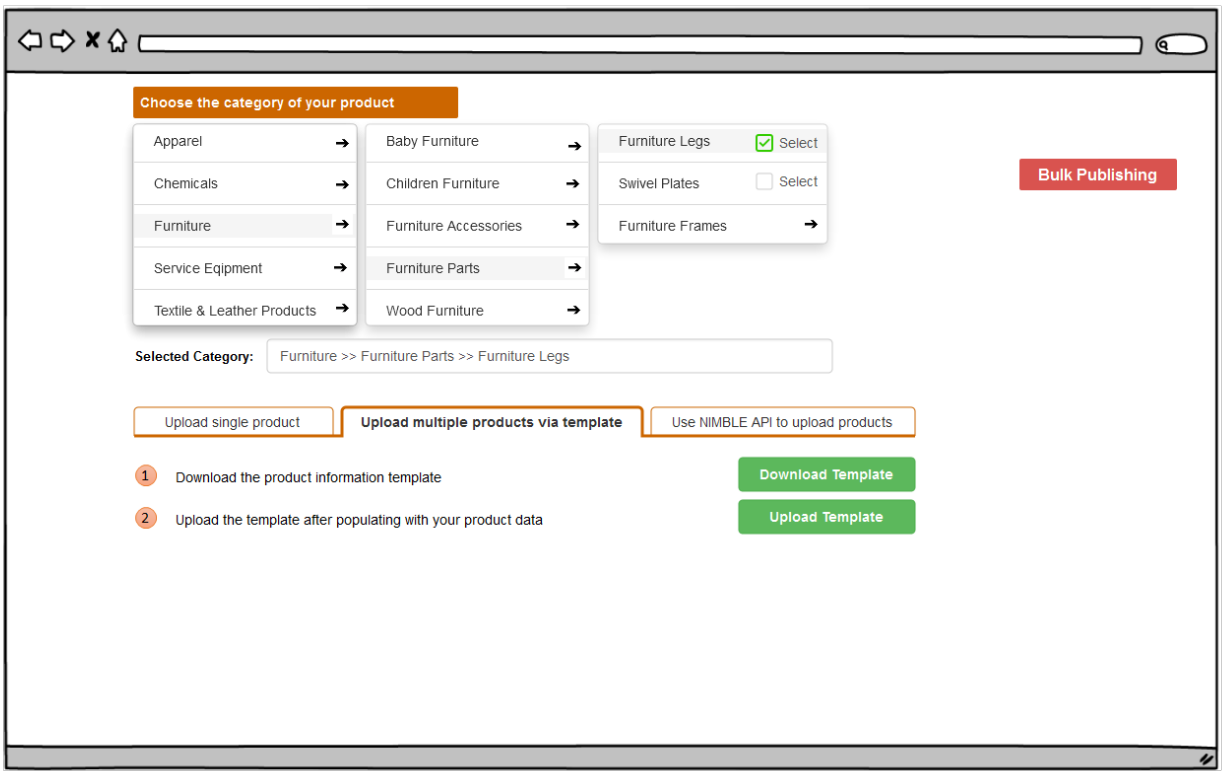
Name: Name

Material: Plastic

Length: 25

Width: 25

Figure 14 Publishing a single product via the Catalogue Registry UI



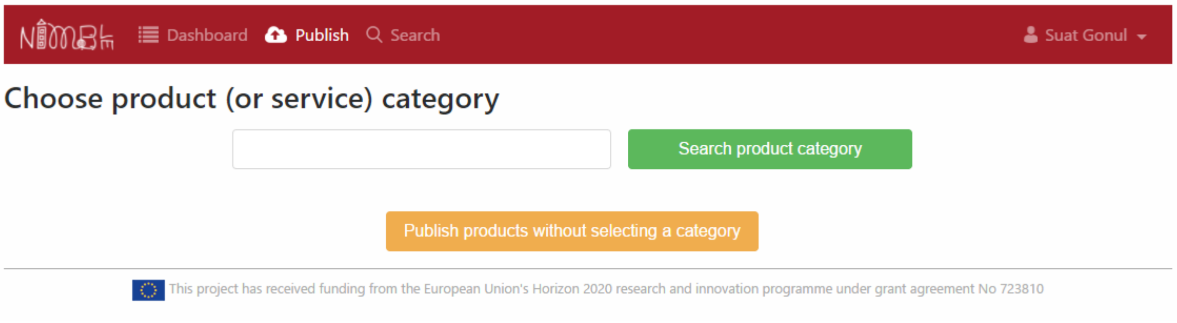
Selected Category: Furniture >> Furniture Parts >> Furniture Legs

1 Download the product information template

2 Upload the template after populating with your product data

Figure 15 Downloading and uploading category templates

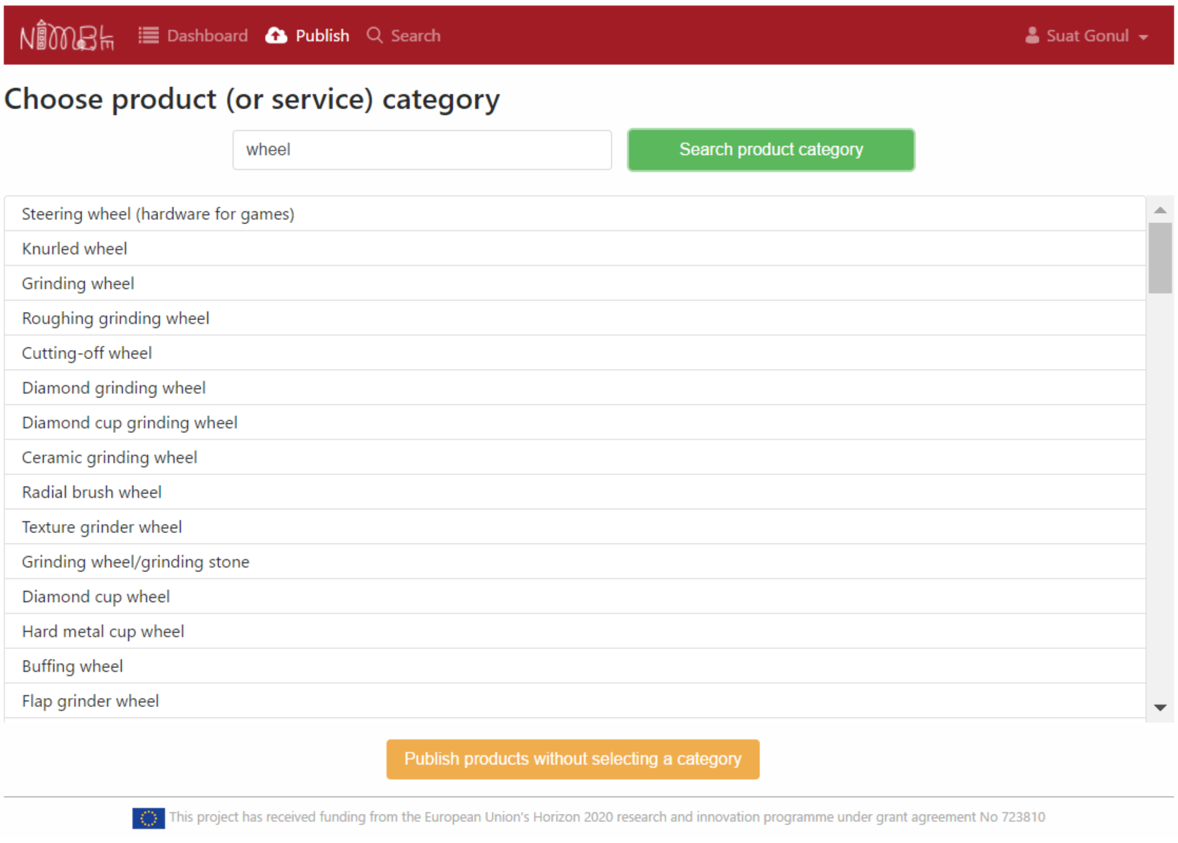
9.4 Annex D – Snapshots from the Catalogue Registry UI



Choose product (or service) category

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723810

Figure 16 Keyword-based category search



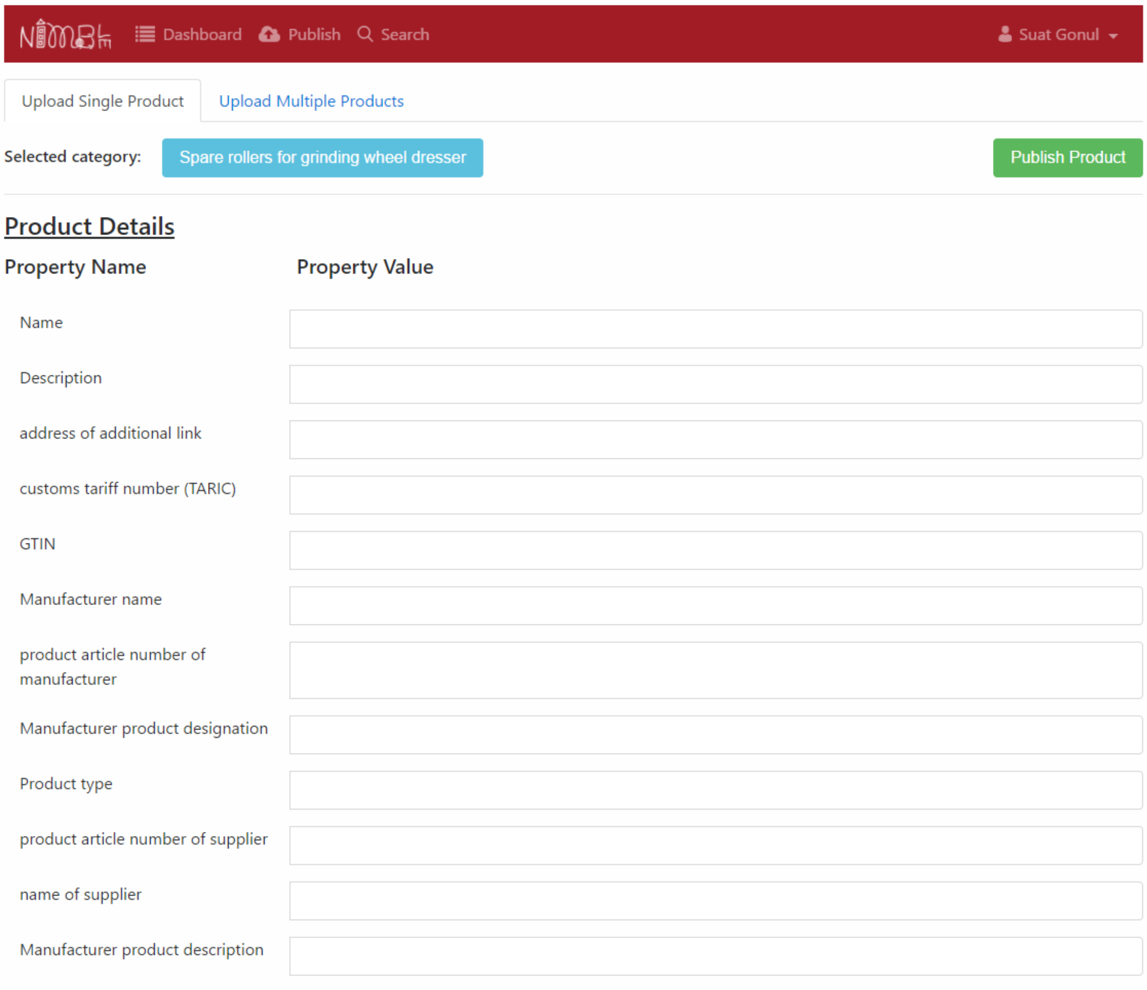
Choose product (or service) category

wheel

Steering wheel (hardware for games)
Knurled wheel
Grinding wheel
Roughing grinding wheel
Cutting-off wheel
Diamond grinding wheel
Diamond cup grinding wheel
Ceramic grinding wheel
Radial brush wheel
Texture grinder wheel
Grinding wheel/grinding stone
Diamond cup wheel
Hard metal cup wheel
Buffing wheel
Flap grinder wheel

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723810

Figure 17 Category search results

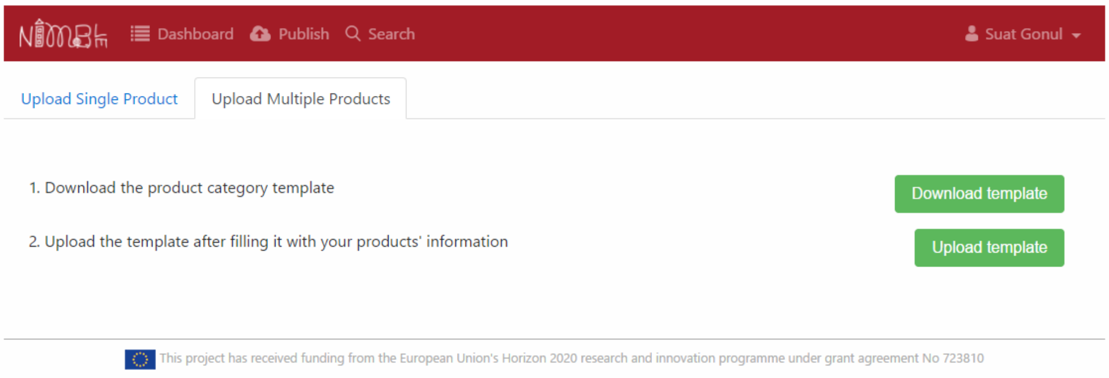


Selected category: Spare rollers for grinding wheel dresser

Product Details

Property Name	Property Value
Name	
Description	
address of additional link	
customs tariff number (TARIC)	
GTIN	
Manufacturer name	
product article number of manufacturer	
Manufacturer product designation	
Product type	
product article number of supplier	
name of supplier	
Manufacturer product description	

Figure 18 Product properties recommended for the select category



1. Download the product category template
2. Upload the template after filling it with your products' information

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723810

Figure 19 Downloading and uploading category templates