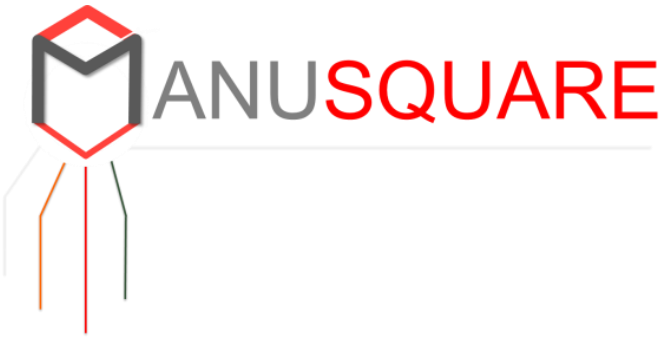
Horizon 2020 – The EU Framework Programme for Research and Innovation
Project Co-funded by the European Commission
Contract number: 761145
Call identifier: NMBP-22-2017
Project Start Date: 1st January 2018



MANUfacturing eco**S**ystem of **QUA**lified **R**esources **E**xchange

D2.2
Domain ontology authoring tool

Dissemination Level	Public
Partners	INNOVA, JPM, TRUDEL, I-COT, I-HUB, CSEM
Authors	Marko Vujasinovic , Alessio Gugliotta
Planned date of delivery	M12 – December 2018
Date of issue	31st December 2018
Document version	Final 1.0

DOCUMENT HISTORY

Version	Issue date	Content and changes	Author
0.1	27.11.2018	ToC	Alessio Gugliotta, Marko Vujasinovic
0.2	04.12.2018	Draft of executive summary and introduction	Marko Vujasinovic, Alessio Gugliotta
0.3	11.12.2018	State of the art reviewed and summarized into § 3	Marko Vujasinovic, Alessio Gugliotta
0.4	15.12.2018	Most of the sections drafted	Marko Vujasinovic, Alessio Gugliotta
0.5	17.122018	First draft completed	Marko Vujasinovic, Alessio Gugliotta
0.6	17.12.2018	Proofreading by the authors. Version shared with consortium and internal reviewers	Marko Vujasinovic, Alessio Gugliotta
0.7	19.12.2018	Incorporated minor revisions by Felix Kurth (CSEM)	Felix Kurth, Marko Vujasinovic,
0.8	20.12.2018	Incorporated suggestions provided by SUPSI – Andrea Bettoni	Andrea Bettoni, Alessio Gugliotta
0.9	20.12.2018	Incorporated revisions provided by Henrique Diogo Silva	Henrique Diogo Silva , Marko Vujasinovic
0.9-1	21.12.2018	Added couple of more screenshots of the tool. Added reference to source code on GitHub. Proofreading.	Marko Vujasinovic
1.0	21.12.2018	Final version ready	Marko Vujasinovic, Alessio Gugliotta
1.0-1	31.12.2018	Quality assurance	Andrea Bettoni

Role	Partner	Person
Reviewer 1	CSEM	Felix Kurth
Reviewer 2	INESC	Henrique Diogo Silva
Quality assurance	SUPSI	Andrea Bettoni

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Acronym	Description
CQ	Competency Questions
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
IRI	Internationalized Resource Identifier
MANU-SQUARE	MANUfacturing ecoSystem of QUAlified Resources Exchange
MODE	MANU-SQUARE Ontology development methodology
MSDL	Manufacturing Service Description Language
MVP	Minimal Viable Product
NAICS	North American Industry Classification System
ODP	Ontology design patterns
OWL	Ontology Web Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
SPIN	SPARQL Inferencing Notation
SUMO	Suggested Upper Merged Ontology
UNSPSC	United Nations Standard Products and Services Code

1 EXECUTIVE SUMMARY

This document presents the results of the extension of MANU-SQUARE Core ontology, developed in Task 2.1, into a formalized domain-specific ontologies of two industrial sectors addressed in the project. These sector are traditional manufacturing industry with a new product development, and textile-cosmetics cross-sector with a by-product-based development of new products. The extension of MANU-SQUARE Core ontology into specific domains was driven by the inputs from the domain experts of the pilot companies including JPM Company, TRUDEL, I-COTTON, I-HUB and CSEM, and scoped to support the MANU-SQUARE business process scenarios, which are defined in Deliverable 1.3. In addition to the description of the developed domain-specific ontologies, this document describes an MANU-SQUARE ontology authoring tool. This tool is provided in the platform on order to facilitate further development, expansion and evolution of the domain-specific ontologies in MANU-SQUARE.

The document starts from a brief review of literature and state of the art in ontology development methodologies and ontology development tools, proceeds to describe a MANU-SQUARE's domain-specific ontology development methodology and to describe the application of this methodology in two demonstration sectors. Then, it presents an MANU-SQUARE's ontology authoring tool, and finally provides concluding remarks and future steps.

The main outcome of this deliverable are the means (that include methodology, design principles and tool) by which a new domain-specific ontologies for MANU-SQUARE shall be developed or existing domain-specific ontology shall be evolved or extended.

This deliverable consists of the following sections:

- § 1 provides an executive summary.
- § 2 briefly introduces the MANU-SQUARE project and the manner in which development of domain-specific ontologies and ontology tool relates to other components within the project.
- § 3 provides an introduction of ontology development process, by the synthesis of the relevant literature and current practices.
- § 4 places an emphasis of MANU-SQUARE-specific ontology development methodology, which builds on a generic ontology development process, but has certain specifics. Those specifics are related to the recommendations of ontology design in MANU-SQUARE, resulting from requirements of the ecosystem.
- § 5 describes the extension of MANU-SQUARE Core ontology concepts, developed in Task 2.1, into a formalized domain-specific taxonomies to support semantic matchmaking in two industrial sectors (traditional manufacturing industry with a new product development; textile-cosmetics industry with a by-product-based processes).
- § 6 details the design and implementation of MANU-SQUARE ontology authoring tool.
- § 7 provides conclusions and the future steps.
- Appendices.

2 INTRODUCTION

The MANU-SQUARE project creates an ecosystem that acts as a virtual marketplace bringing the available production capability and unused capacity, as well as other virtual and physical assets, closer to the production demand to obtain the optimal match between the supply and demand (i.e. between manufacturers and customers), as shown in Figure 1. The MANU-SQUARE ecosystem, based on the optimized matchmaking between production supply and demand, will provide two main advantages:

- The reintroduction and optimization in the loop of unused production capacity and production/business potential that would otherwise be lost;
- The rapid and efficient creation of local distributed value networks for innovative providers of product services;

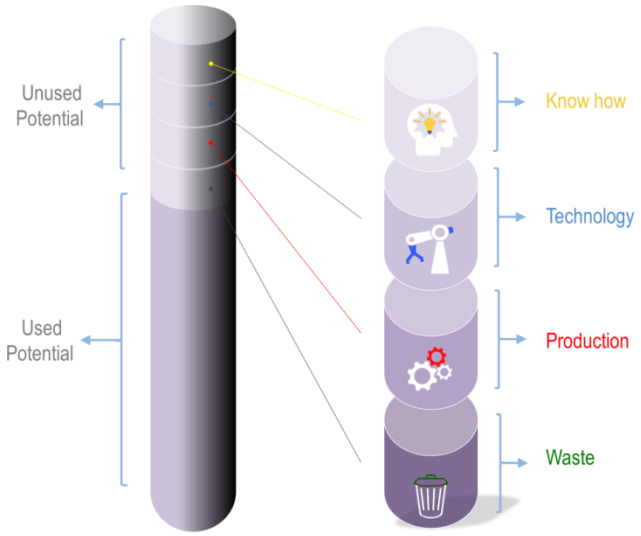


Figure 1 Composition of the unused potential

Therefore, MANU-SQUARE establishes an ecosystem that is organized to match the needs of customer with the availability of suppliers in terms of their four key production-related resources: 1) know-how, 2) technology, 3) capacity and 4) by-products (waste).

In general, a manufacturing company that uses MANU-SQUARE may have a role of supplier (seller) or customer (buyer). In the case of the former, the manufacturing company uses MANU-SQUARE to create its respective profile that advertises its manufacturing services in the terms of capabilities, capacities, know-how, products, and by-products. In the case of the latter, the manufacturing company uses the platform when it requires to engage with the MANU-SQUARE ecosystem to fulfil a manufacturing-related need, such as an additional production capability. The platform performs the search for the optimal matching on a wide number of possible suppliers from the MANU-SQUARE register, using a sophisticated criterion that ensures high level of quality and reliability of suppliers, reduction of costs and short time to close the business transaction. The ecosystem allows discovery also for resources other than production capability, e.g. available production hours or tangible assets, with the aim to identify and exploit unexpected synergies between participants and to promote the mutual interaction of diverse industries, also within different value networks, for beneficial reuse of competences and flows.

The main objectives of the project are:

1. Make European unused manufacturing capacity emerge towards its reintegration in the loop and the creation of local efficient value networks.
2. Support innovative SMEs and start-ups in finding the optimal suppliers to transform their business ideas into new product-services.

3. Seamlessly involve actors all along the entire value network including consumers for cross-fertilization of product-service solutions and underlying technologies.
4. Coordinate the whole ecosystem towards a better use of resources and a more sustainable European manufacturing.

To achieve these objectives, MANU-SQUARE platform offers a set of integrated advanced features including a production capabilities matchmaking, by-products matching, suppliers and customers reputation management, innovation management, suppliers' assessment, sustainability assessment, and request-for-quantitation management. The platform will use a blockchain technology for secured execution of supply chain scenarios and will use a semantic infrastructure for formal description of MANU-SQUARE resources, resources 's discovery and knowledge sharing based on their formalized semantics.

2.1 Aim, scope and inter-tasks relationships of Task 2.2

Task 2.2 is one of the tasks of MANU-SQUARE's Work Package 2 (Ecosystem ontological representation). WP2 provides: (1) a semantic infrastructure for MANU-SQUARE platform (as shown in Figure 2); (2) tool-supported development of ontological representation of manufacturing concepts and their taxonomies, (3) application-specific models and inference rules.

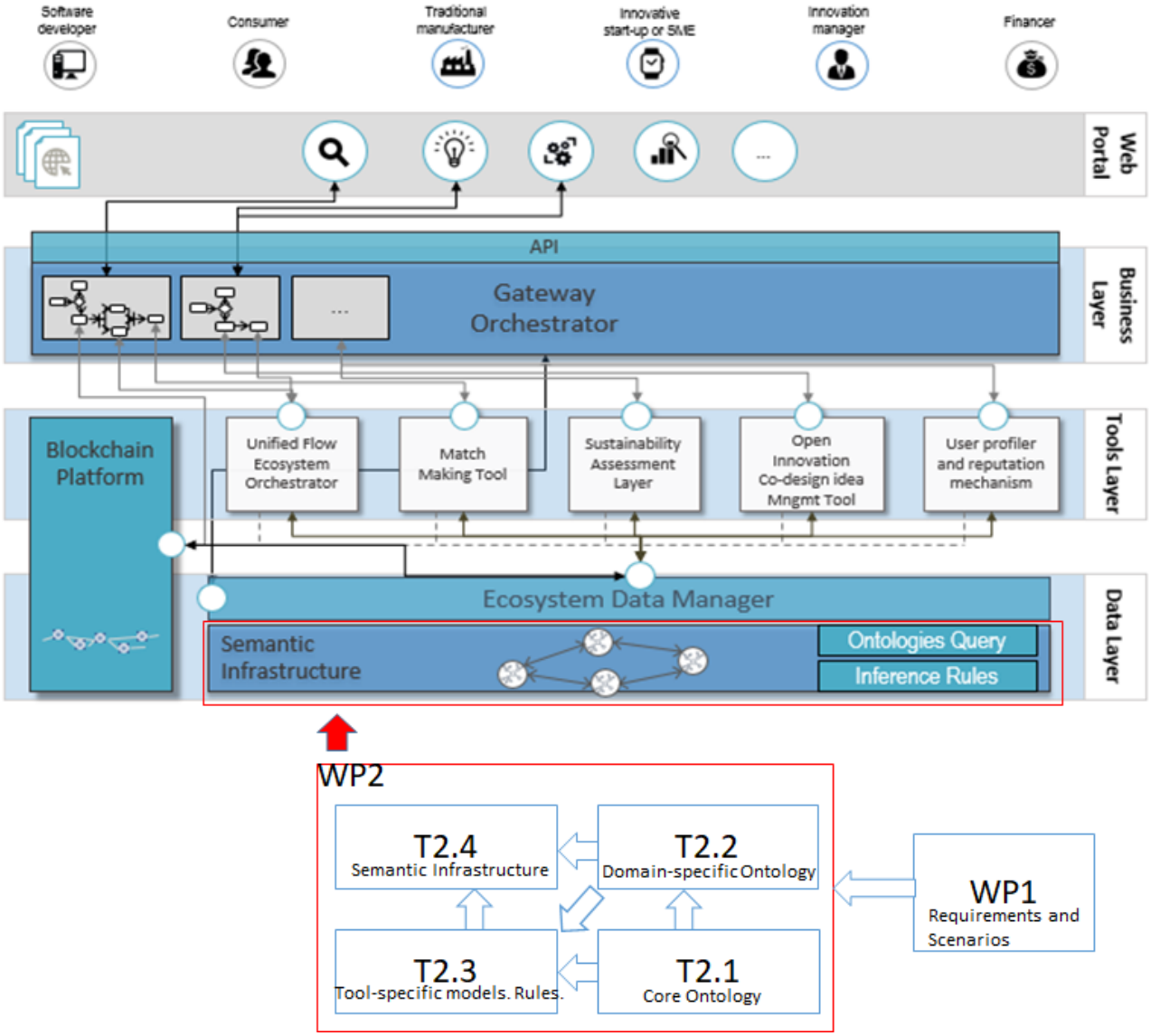


Figure 2 High level schema of MANU-SQUARE architecture and WP2 role within it

Task 2.1 provided a MANU-SQUARE Core Ontology (Ecosystem Data model), which has been reported in D2.1. This Core ontology, or Code Concepts model, is a set of generic concepts that are shared, within MANU-SQUARE ecosystem, across multiple manufacturing sectors and applications dealing with (1) manufacturing capability and capacity, (2) manufacturing innovation idea, (3) manufacturing sustainability assessment, (4) request for quotation or ordering, and (5) ecosystem stakeholders reputation. This core ontology, and mainly its part related to the Factory model, forms the basis for formal and unambiguous semantic descriptions of production resources in MANU-SQUARE. It is a common upper vocabulary for descriptions of core manufacturing concepts - Process, ProcessType, Energy, EnergyType, Item, Component, Capability, Resource, Supplier, and others.

Task 2.2 provides vertical extension of MANU-SQUARE Core Ontology (Ecosystem Data model) into domain-specific ontologies, by capturing domain-specific taxonomies of process types, equipment types, capability types, product types, material types. For example, it captures a taxonomy of equipment types in the machining sector (e.g. Milling Machine, Drilling Machine, etc.). The conceptual scope of these domain-specific ontologies is narrowed to the scope of two MANU-SQUARE demonstration scenarios: (1) machining sector scenario that includes a new product development use-case - production of an automated guided vehicle solution; (2) textile and cosmetics sector scenario where one of the pilot companies provides a by-product to other pilot company for the development of new products. In addition to the development of these two sector-specific ontologies, another T2.2 outcome is an ontology authoring tool to facilitate ongoing development, expansion and evolution of the MANU-SQUARE sector-specific ontologies, and creation of ontologies for new vertical sectors. The tool is based on an interactive ontology development process, with controlled semi-automated knowledge elicitation (entity/concept extraction), and with RDFS/OWL representation and verification of ontologies.

Task 2.3 further enhances the core model of Task 2.1 with ontological models containing the propositions required for supporting the functioning of the service-providing applications, which are being developed in WP4. These application-specific models contain specific set of information the applications use for providing their functions. For instance, the application-specific models describe the impact of production processes expanding the generic concept of Process described in the MANU-SQUARE Core Ontology in order to allow the execution of environmental assessment engine (Task 4.3). In addition to that, Task 2.3 defines a set of inference rules that allow for the finding and inferring of new facts by reasoning on the data present in the semantic database. Rule-based inference is the basis for discovery and recommendation of new production opportunities residing in the ecosystem (e.g. An innovation manager can define a rule that specifies that a certain waste type, produced by a process, is actually a by-product, thus, a possible raw material for other products. Hence, new production opportunities may emerge).

A semantic infrastructure (Task 2.4) is expected to offer the following capabilities: RDF data store and querying over structured domain ontologies or knowledge bases, semantic inferences based on area, service-specific rules, support for semantic matchmaking and discovery of MANU-SQUARE resources including support for semantics descriptions of manufacturing services, capabilities, products, and know-how. The semantic infrastructure, thus, shall store, describe and maintain the knowledge that is necessary to the other platform tools to implement added-valued, automatic processes.

2.2 Domain-specific ontologies and inference rules use in MANU-SQUARE

The matchmaking of MANU-SQUARE resources (manufacturing services, capabilities, by-products, know-how) is the main service that exploits the domain-specific ontologies. The matchmaking is the functionality that compares the data/parameters that describe the needed resources, as defined by a customer, with description of available resources, as entered by suppliers.

Matchmaking on the traditional manufacturing sourcing platforms[1] is currently based on a text-based search algorithms that perform literal matches of the search words or variants of them with words contained in the textual descriptions of production capabilities or capacities. The text-based search on informal textual descriptions is not a good approach if the users want results that undoubtedly match the search request.

In contrast to that, MANU-SQUARE's matchmaking tool takes the semantic-based matchmaking approach, which has the potential to provide a higher rate of search precision when compared to text-based search methods. The semantic-based matchmaking works on a semantic, formal, unambiguous and structured description of production capabilities/capacities and utilizes a domain knowledge with semantic relationship between concepts. The purpose of domain specific ontologies is to provide a common, formal and structured vocabulary for these descriptions.

Inference rules will also contribute to the added-valued, automatic processes in the platform. An inference reasoner derives additional information from the descriptions of MANU-SQUARE resources, driven by the rules, and turns this additional information into new descriptions and new knowledge in the ecosystem. The inference rules may make explicit previously not stated (unknown) facts about resources and their relationships. For example, at one future point there can emerge, based on novel research and technological advancements, a completely new technology that turns certain waste into a by-product that can be further used as an input for certain production processes. This new knowledge in the domain, which was unknown when a platform was deployed, can be later on expressed by a simple rule (if a process outcome is a waste, and waste type is X, classify all such outcomes as by-products of type Y, and classify instance of type Y as process inputs of process whose type is Z). By executing such rule on the database, new links between resources will be created and new production opportunities may emerge.

2.3 Current status

The expected outcome of Task 2.2 is two-fold. First is a domain-specific extension of MANU-SQUARE core concepts to facilitate execution of the two MANU-SQUARE's demonstration scenario (scenarios are outlined in D1.3). Second one is an ontology authoring tool that should help the platform managers and domain experts to: (a) facilitate development and evolution of the domain-specific extension of core concepts when platform further expands within existing or enters into new sectors, and (b) to ease the provisioning of area-specific models and inference rules.

The current status is as follows:

Domain-specific ontologies are built, encoded in OWL/RDFS and deployed in a semantic database. The conceptual coverage of domain-specific ontologies has been narrowed to the current scope of two MANU-SQUARE demonstration scenarios. The domain knowledge elicitation process involved all the partners/experts from the demonstration scenarios, namely JPM, TRUDEL, I-COTTON, I-HUB, CSEM, and SUPSI for the tools-perspective. The domain knowledge elicitation has been done using a questionnaire and interviews. The questionnaire was structured to collect information about needed taxonomical extensions of process types, products types, by-product types, material type, human and equipment capability types, KPI types, and other key concepts and properties in two respective domains (see Appendix A for the questionnaire details). The completed questionnaires was analysed by ontology development experts (INNOVA) and further discussions with domain experts were held to establish a common meaning of concepts and to clear any ambiguities in the domain ontologies. Another questionnaire was prepared for the tool developers, in order to collect the competency questions that domain-specific ontologies should be able to answer (see Appendix B for the questionnaire details).

During the domain-specific ontology conceptualisation, the ontology experts have investigated existing RDFS/OWL ontologies from manufacturing or textile-cosmetic domain that could be reused and integrated into MANU-SQUARE domain ontologies. No complete out-of-the-box solutions exists, but there are relevant standard classifications such as

[1] Examples of traditional platforms considered here are online manufacturing sourcing platforms, such as ThomasNet, GlobalSpec, where descriptions of suppliers are unstructured and informal, provided using a free-text without controlled and formalized vocabularies.

an industry classification (e.g. NAICS-North American Industry Class System), or classification of products and services (e.g. UNSPSC - United Nations Standard Products and Services Code) that could be reused and integrated into MAU-SQUARE. More specifically, such standardised classifications could be translated into OWL taxonomies and integrated accordingly with MANU-SQUARE domain ontologies. However, this consists future work as it goes beyond the project scope. At the moment, MANU-SQUARE domain ontologies only capture demonstrative portions of taxonomies of key concepts in two manufacturing sectors relevant for the platform demonstration. As previously mentioned, these taxonomies are built by the knowledge elicitated from the project partners. In addition, there are reused parts of a MSDL ontology (Ameri and Dutta 2006)[2] related to taxonomies of manufacturing services, machining sector equipment, products, certifications and industries.

A first version of an ontology authoring tool to support manual development of ontologies using structured templates of taxonomic schemes for process types, products types, by-product types, material type, human and equipment capability types has been released. The tool is described in § 6. Notice that the current release does not completely provide the support for semi-automated knowledge elicitation (based on the natural language processing, entity extraction, and concept recognition), which requires additional interactions and specifications with domain experts. Tool, however, provides an interface to user to paste a short text to be evaluated by a third-party Entity Extraction API - Dandelion API (https://dandelion.eu/docs/api/datatxt/nex/v1/), as shown is § 6 – Figure 26. Dandelion API is a named entity extraction API that is able to detect concepts from the short text using knowledge from DBpedia (https://wiki.dbpedia.org/) and Wikipedia (https://www.wikipedia.org/). Further, the tool does not yet fully provide for provisioning of area-specific models and inference rules which are expected to be developed as part of Task 2.3 and then accordingly incorporated into the tool. The missing features will be provided in a next release of the tool, as part of Task 2.3 and Task 2.4 expected developments. The tool provides ontologies represented using RDFS/OWL language and verified with a built-in reasoner.

2.4 Outline

This deliverable is organized into the eight sections. First section provides an executive summary. § 2 briefly introduces the MANU-SQUARE project and the manner in which development of domain-specific ontologies and ontology tool relates to other components within the project. § 3 provides an introduction of ontology development process, by the synthesis of the relevant literature and current practices in similar domains. § 4 places an emphasis of MANU-SQUARE-specific ontology development methodology (MODE), which builds on a generic ontology development process, but has certain specifics related to the ontology design recommendations according the specific requirements of the ecosystem. § 5 describes the extension of MANU-SQUARE Core ontology concept used in Task 2.1, into a formalized domain-specific taxonomies to support semantic matchmaking in two industrial sectors addressed in the project (traditional manufacturing industry with a new product development; textile & cosmetics industry with a by-product-based new product development). § 6 details the design and implementation of MANU-SQUARE ontology authoring tool. Finally, § 7 provides conclusions and some future steps. Appendices A and B provide an insight into questionnaires used for the domain knowledge elicitation. Appendix C provides examples of SPARQL queries that exploit MANU-SQUARE Core ontology and the domain-specific taxonomies to retrieve the described manufacturing resources.

[2]MSDL OWL file is available from http://infoneer.wp.txstate.edu/ontology-download/msdl-ontology/

3 ONTOLOGY DEVELOPMENT: STATE OF THE ART

3.1 What are ontologies?

Ontology, as a formal conceptual model, is a formal explicit description of concepts (i.e. classes) in a domain of discourse, properties of each concept describing various features and attributes of the concepts, and restrictions on the properties (Noy, et.al 2001). Thus, an ontology includes machine-interpretable definitions of concepts in the domain and relations among them. An ontology together with a set of individual instances of classes constitutes a knowledge base (Noy, et.al 2001).

Ontologies allow the resources to be semantically enriched, which is a pre-condition to provide new, advanced services over the web, such as the semantic search and retrieval of resources (De Nicola, et al 2005). And this is the main reason why MANU-SQUARE services builds on the semantic technology and ontologies.

Ontologies, as a formal and explicit description of concepts in a domain, are implemented (i.e. captured, represented) using ontology languages. The ontology languages are always formal languages, with concrete syntax and semantics of language constructs. They are used to construct and validate the ontologies, supported by the ontology development tools. Ontology languages, thus, allow the formal representation and encoding of domain, formalization and axiomatization of the domain concepts, and often include reasoning rules that support knowledge processing and inference. According to the ontology representation-related requirements of MANU-SQUARE platform (discussed in Deliverable 2.1), the chosen ontology languages for the platform are RDFS and OWL. Both are based on Resource Description Framework (RDF)[3] and provide needed expressiveness and inference capability.

Ontologies can be of different types as we can evaluate them from different perspectives e.g. the level of abstraction of concepts, the conceptual coverage of the ontology, richness of concepts axiomatization. For example, there are **foundation ontologies** such as DOLCE, SUMO, or OpenCyc that formally axiomatise domain independent (upper) set of concept. Foundational ontologies are developed to serve as upper ontologies that cover (bridge) every domain in a highly generalized way. Foundational ontologies are not suitable for manufacturing domains, as discussed by (Usman, et.al. 2013). Then, there are core ontologies of specific domains, domain-specific ontologies, vocabularies, taxonomies. A **core ontology** provide a set of generic concepts whose semantics are shared across multiple, but not all, domains. A **domain-specific ontology** provide a set of real-world concepts whose semantics is shared across single domain e.g. manufacturing services description, description of product life-cycle or process/discrete manufacturing factory. There are application specific ontologies that include concepts in the application specific scope. Further, **vocabularies** are either core or domain ontologies but without any axiomatization of the ontology concepts and without semantic entailment relationships between concepts. Finally, a **taxonomy** represent the formal sub-class hierarchical structure of classes or types of objects within a domain. A taxonomy is created by grouping things in a domain into categories and sub-categories. Often sub-categories are formed several levels deep (Pieterse, et al. 2014).

3.2 Ontology Development Methodologies

The scientific community has not yet reached a consensus on one or more standard methods for building large-scale ontologies. There are many reasons for this, but one of the main one is that mostly methodologies were applied for developing ontology for a project, which does not unveil much insight to encourage others to adopt it (Rizwan, et al, 2013).

As reported by De Nicola, et al (2005), the first contributions to ontology building methods are due to Gruber (1993), Gruninger and Fox (1995), Uschold and King (1995), and Uschold and Gruninger (1996). Gruber's work discusses basic

[3] The Resource Description Framework (RDF) is a W3C standard for describing (Web) resources and their relationships. RDFS (the Resource Description Framework) extends RDF by providing additional language constructs that allows to define the classes, properties, taxonomies in domain, in order to allow semantically richer descriptions of resources and domain itself. Finally, The Web Ontology Language (OWL) extends RDFS and allows for expressing further schema definitions in RDF.

ontology design criteria. These are clarity, coherence, extendibility, minimal encoding bias and ontological commitment. Gruninger and Fox (1995) provide a skeletal methodology for ontology building, while a method based on competency questions is presented by Uschold and King (1995). Ushold and King were the first who felt the need to propose a methodology for the purpose of developing ontologies, based on the experience for developing the Enterprise Ontology, as reported in (Rizwan, et al, 2013). The Gruninger and Fox (1995) methodology first focuses to capture the ontology requirements by means of informal description. Later, informal description is transformed to formal language. Motivation scenarios are used to capture the informal intended semantics which should be introduced in the ontology. From these motivation scenarios evolve the competency questions, they are considered as expressive requirements which the ontology should answer (Gruninger and Fox, 1995)

Competency question (CQ, in short) is an important technique for elicitation of ontology requirements. They determine the scope of the ontology and requirements of intended, content-related uses of ontology such as capturing, retrieval, inference and validation of domain knowledge or domain information. CQ are introduced in (Grüninger and Fox 1995-a) as requirements that form a basis for a rigorous characterization of the problems that the ontology is able to solve. These questions serve as the litmus test later: Does the ontology contain enough information to answer these types of questions? Do the answers require a particular level of detail or representation of a particular area? The CQs are just a sketch and do not need to be exhaustive (Noy and McGuinness, 2001).

Fernández et al. (1997) proposed a complete ontology development process, named **METHONTOLOGY**. Their ontology development process is composed by these phases: specification, conceptualization, formalization, integration, implementation, maintenance. The life cycle of ontologies is based on evolving prototypes. In METHONTOLOGY, other activities, like control, quality assurance, knowledge acquisition, integration, evaluation and documentation are carried out simultaneously with the ontology development activities (De Nicola, et al, 2005).

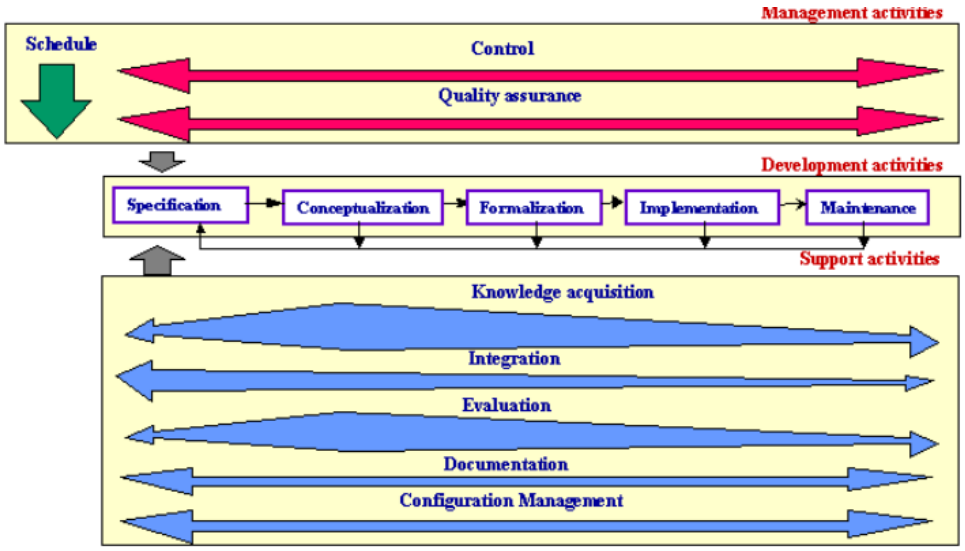


Figure 3 Activities in the ontology development proposed by METHONTOLOGY (source: Corcho et al, 2005)

Sure et al. (2004) propose **On-To-Knowledge**, an ontology development process consisting of feasibility study, kick-off, refinement, evaluation, application and evolution phases.

SENSUS-based methodology was used to build SENSUS - an ontology for use in natural language processing that was developed at the ISI (Information Sciences Institute) to provide a broad-based conceptual structure for developing machine translators. SENSUS has more than 50,000 concepts organized in a hierarchy, according to their level of abstraction. It includes terms with both a high and a medium level of abstraction, but, generally speaking, does not cover terms from specific domains (Fernández-López, 1999), (Swartout et al., 1996)

As reported in Fernández-López (1999), in SENSUS methodology, when an ontology is to be built in a particular domain, the following steps are taken (Swartout et al., 1996): Step 1. A series of terms are taken as seed: Step 2. These seed terms are linked by hand to SENSUS; Step 3. All the concepts in the path from the seed terms to the root of SENSUS are included; Step 4. Terms that could be relevant within the domain and have not yet appeared are added; Step 5. Finally, for those nodes that have a large number of paths through them, the entire sub-tree under the node is sometimes added, based on the idea that if many of the nodes in a sub-tree have been found to be relevant, then the other nodes in the sub-tree are likely to be relevant as well.

101 method is yet another methodology for developing domain ontologies (Noy and McGuinness, 2001). It is iterative in nature and based on some fundamental rules which assist in making design decisions during ontology development. The guide sequentially covers all the phases of ontology development, including complex issues related to defining class hierarchies and properties of classes and instances. Noy and McGuinness (2001) propose Knowledge-Engineering Methodology with these steps: Step 1. Determine the domain and scope of the ontology (usually with competency questions); Step 2. Consider reusing existing ontologies; Step 3. Enumerate important terms in the ontology; Step 4. Define the classes and the class hierarchy; Step 5. Define the properties of classes—slots; Step 6. Define the facets of the slots; Step 7. Create instances.

UPON is ontology development methodology derived from the Unified Software Development Process (De Nicola et al., 2005). The methodology adapts the Unified Process (UP) paradigm of software engineering and the Unified Modelling Language (UML). Ontology development using UPON consists of cycles, phases, iterations and workflows, as it follows the UP paradigm. The use-case driven, iterative and incremental nature of UPON makes it unique from other processes, respectively for software and ontology engineering (Nicola et al., 2005). UPON is use-case driven in that it aims at producing an ontology with the purpose of serving its users, both humans and automated systems. UPON was applied in the context of the Athena Project for building an ontology of eProcurement.

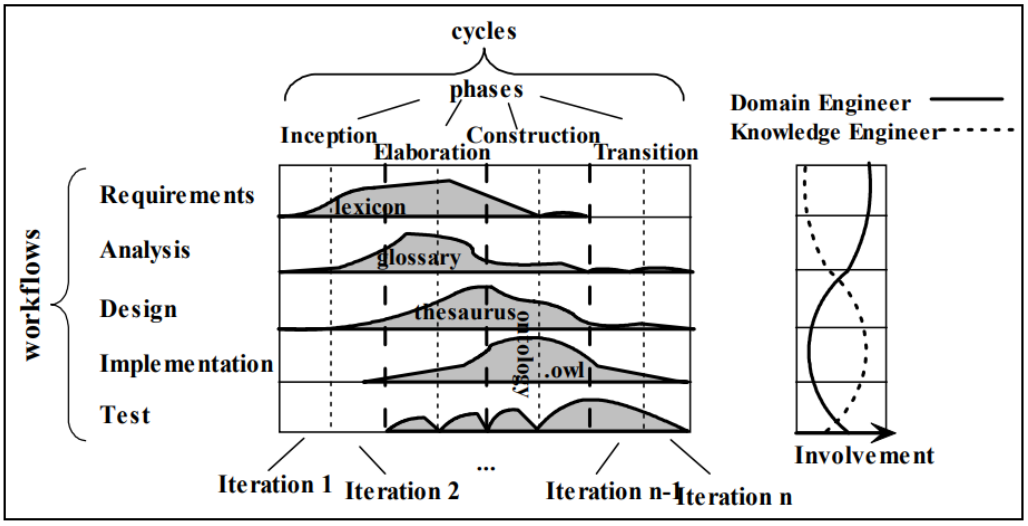


Figure 4 UPON Ontology Development Framework (source: De Nicola et al., 2005)

In UPON, the first iterations (inception phase) are mostly concerned with capturing requirements and partly performing some conceptual analysis. During subsequent iterations (belonging to the elaboration phase) analysis is performed and the fundamental concepts are identified and loosely structured. Most of the design and implementation workflows pervade iterations in the construction phase. In the construction phase some additional analysis could be still required aiming at identifying concepts to be further added to the ontology. During the final iterations (transition phase), testing is heavily performed and the ontology is eventually released.

The Requirements workflow in UPON aims at (i) determining the domain of interest and the scope, (ii) defining the purpose or motivating scenario, (iii) writing a storyboard, (iv) creating the application lexicon, (v) identifying the

competency questions, and (vi) use-case identification and prioritization. The Analysis Workflow has these activities (i) consideration of reuse of existing resources: (ii) identification of relevant terms (domain lexicon); (iii) modelling the application scenario using UML diagrams, (iv) building the glossary. The Design Workflow is (i) categorisation of concepts (ii) refinement of the concepts and their relations. The outcome of this step is a UML class diagram. The Implementation Workflow formalize the ontology in a language and implements it in terms of components. Finally, the Test Workflow allows to verify that the ontology correctly implements its requirements.

NEON project systematises the six most common scenarios that may unfold during the ontology development, as illustrated in Figure 5 (Suárez-Figueroa, 2012).

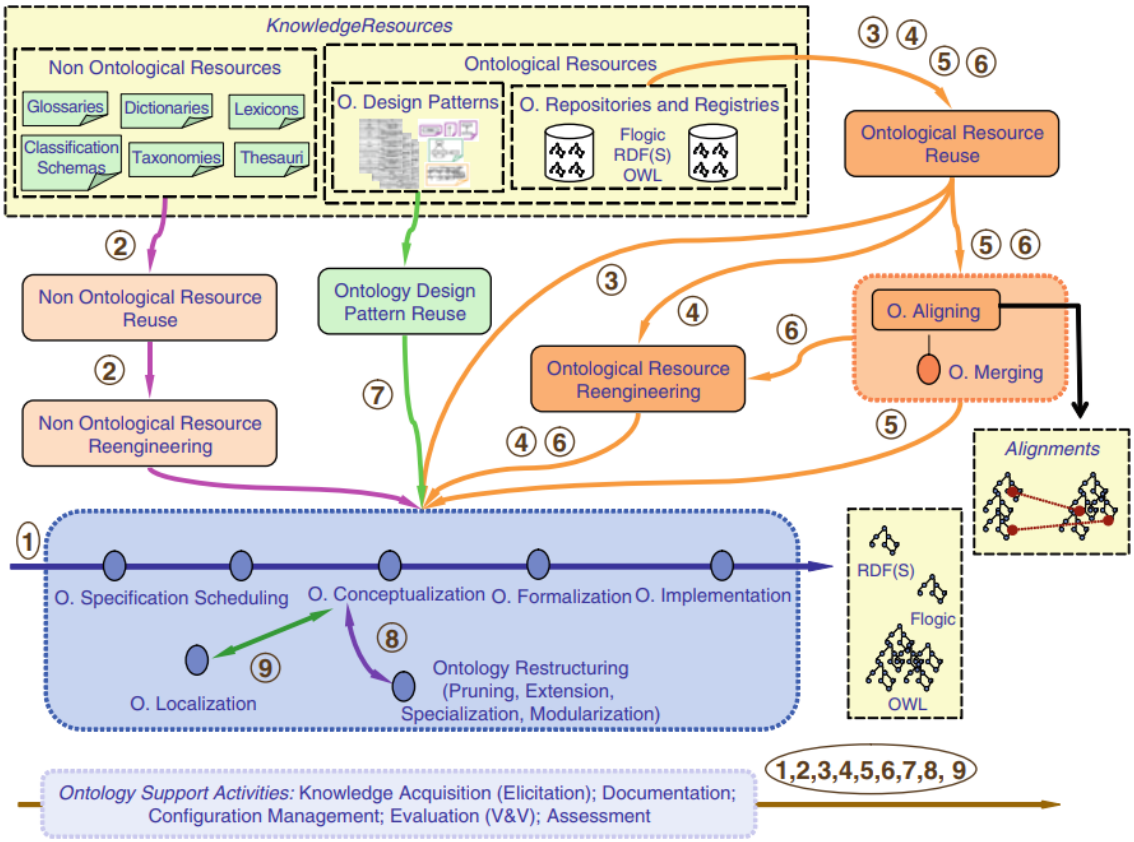


Figure 5 Scenarios for building ontologies and ontology networks (source: Suárez-Figueroa, 2012)

- Scenario 1: From specification to implementation. The ontology network is developed from scratch, that is, without reusing available knowledge resources.
- Scenario 2: Reusing and re-engineering non-ontological resources. This scenario covers the case where ontology developers need to analyse non-ontological resources and decide, according to the requirements the ontology should fulfil which non-ontological resources can be reused to build the ontology network.
- Scenario 3: Reusing ontological resources. Here, ontology developers reuse ontological resources.
- Scenario 4: Reusing and re-engineering ontological resources. Here, ontology developers both reuse and re-engineer ontological resources.
- Scenario 5: Reusing and merging ontological resources. This scenario unfolds only in those cases where several ontological resources in the same domain are selected for reuse and when ontology developers wish to create a new ontological resource from two or more ontological resources.
- Scenario 6: Reusing, merging, and re-engineering ontological resources. This scenario is similar to Scenario 5; however, here developers decide not to use the set of merged resources as it is, but to re-engineer it.
- Scenario 7: Reusing ontology design patterns (ODPs). Ontology developers access ODPs repositories to reuse them.
- Scenario 8: Restructuring ontological resources. Ontology developers restructure (modularizing, pruning, extending, and/or specializing) ontological resources to be integrated in the ontology network being built.

- Scenario 9: Localizing ontological resources. Ontology developers adapt an ontology to other languages and culture communities, thus producing a multilingual ontology.

Knowledge acquisition, documentation, configuration management, evaluation, and assessment should be carried out during the whole ontology development, that is, in any scenario used for developing the ontology.

In a summary, in all the above mentioned methodologies, the ontology development process starts from the requirements, goes over the conceptualisation and implementation of ontology, and ends with evaluation and testing. Requirements identify the purpose of ontology and its scope. Implementation may involve activities of capturing (coding) ontology in an ontology language, integration with other ontologies, and reusability of ontological resources. Before the implementation takes place there can be activities of defining a glossary of terms and conceptualization. Evaluation should detect possible errors in ontology (unhallowed usage of ontology language constructs) and inconsistences in the captured ontology.

3.3 Ontology Design Patterns[4]

An ontology development should follow a methodological and structured process that leads to the uniformity and consistency in the design of ontology. If the design does not follow a strict design principles/recommendations, the ontology can become very complex, hard to read, refactor, maintain, debug, evolve, and hard to map to other ontologies. Not carefully managed changes in ontology may produce hard-to-debug inconsistencies in already captured knowledge.

To illustrate a situation when “freedom” in ontology design may lead to issues, let consider an example of modelling of descriptive "features" (or, “qualities", "attributes" or "modifiers") of things captured as ontology concepts (Rector, 2005). There are at least two approaches for modelling the “features”: (#1) as individuals whose enumeration make up the parent class representing the feature; (#2) as disjoint classes which exhaustively partition the parent class representing the feature (Rector, 2005). Following both of these recommendation for addressing essentially same set of requirements may end-up in an inconsistent design throughout the ontology. The inconsistent design may cause additional efforts in the ontology activities such as ontology refactoring or querying. Obviously, a query for retrieving the features in approach #1 wouldn’t work over the solution using approach #2., and/or invalid ontology mappings, or a need for a complete remapping. The ontology design freedom is undesired in the applied ontology engineering.

Thus, a domain ontology development needs to be approached in a systematic way, especially large-scale domains, such the MANU-SQUARE domain where ontology development might be a collaborative process that involves many experts. The key to success is to adopt and apply the same modelling principles. Hence, a possible solution is to establish a domain-specific ODPs (ontology design patterns, or, ontology design rules/recommendations), which would be used and strictly respected in the domain ontology development. ODPs are reusable components and design recommendations for capturing recurring ontological content and structures in a uniform and consistent way.

Design patterns have emerged as engineering artefacts in many engineering disciplines, most notably in software engineering. According to Gamma et al. (1995), “a design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem ... it describes the problem, the solution, when to apply the solution, and its consequences.”

Because of the enormous success of design patterns in software engineering, researchers began using the idea of design patterns for the ontology engineering. In particular, W3C Semantic Web Best Pratices and Deployment Working Group introduced ontology design patterns as “best practice" modelling solution in OWL for certain modelling situations (W3C, 2005). Gangemi (2005) introduced Conceptual ODPs to capture foundational and core domain concepts to provide reusable solutions “for solving design problems for the domain classes and properties that populate an ontology. Aranguren (2008) with his colleagues, presented ODPs as best-practice solutions to logical modelling problems leading to better use, greater expressivity, and greater rigor when using OWL for ontologies in bio-science domains. Kulvatunyou, Lee, and Ivezic (2013) proposed application of ODPs for canonicalization of different proprietary

[4] Some portions of this section are taken from (Vujasinovic, et al 2015).

manufacturing service capability (MSC) models and presented the approach with illustrative examples of ODPs for manufacturing. This is illustrated in Figure 6 with a Service pattern.

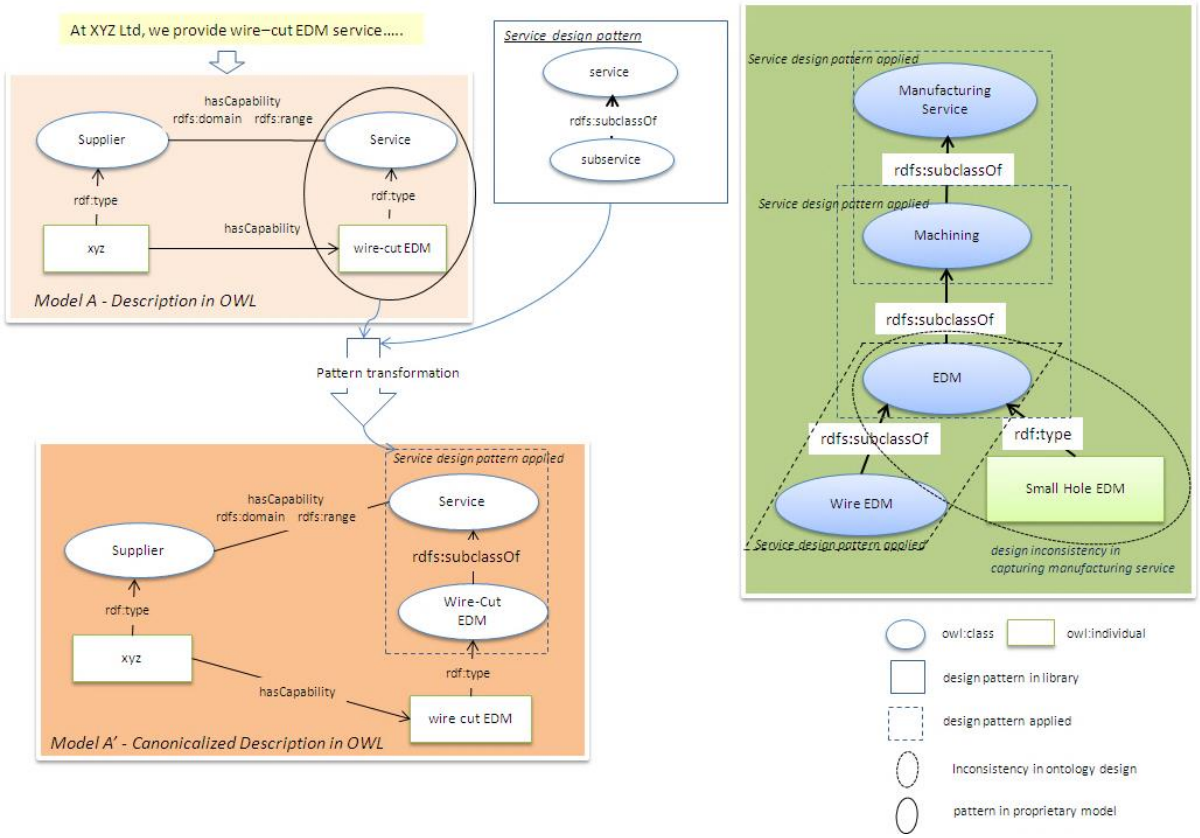


Figure 6 Application of an ODP in semantic modelling of MSC information (source: Vujasinovic, et al. 2015)

ODP classification is provided in Suarez-Figuero *et al.* (2012) and Presutti *et al.* (2008). According to those researchers, ODPs can be grouped into different types, with each type addressing different aspect in ontology development, as outlined below:

- Structural ODPs include Logical and Architectural ODPs. According to Suarez-Figueroa *et al.* (2012), Logical ODPs are compositions of logical constructs of an ontology language. They are useful in solving certain design problems when the ontology language does not directly support certain logical constructs. For instance, *N-Ary Relationship ODP* is a Logical ODP for capturing n-ary relations using OWL, which allows only binary properties. Architectural ODPs are compositions of Logical ODPs and define overall shape of ontology such as taxonomical organization. Logical ODPs are not explicity bound to any domain-specific vocabulary; they are abstract description of ontology concepts, relations or axioms, and they have an empty signature[5].
- Content, sometimes called Conceptual, ODPs are ontology components that capture foundational, core, or domain-specific concepts and their features. As such, they can provide ready-to-use and reusable ontology components for modelling and capturing ontology content. Content ODPs are explicitly bound to a vocabulary for a specific or universal domain. Thus, they have no empty signature. For example, *Place ODP*, *PeriodicInterval ODP* or *Person ODP* from NeOn library are Content ODP.
- Then, there are Correspondence, Reasoning, Presentation and Lexico-syntactic types of ODPs. Correspondence ODPs include Re-engineering and Alignment ODPs. The former provides solutions for transformation of RDBMS or XML models into ontologies; the latter provides solutions for creating semantic associations between ontologies. Reasoning ODPs, such as *Normalization ODP* in Manchester ODP Library

[5] A design pattern signature is a graph representation of the design pattern with nodes and edges either named or not. A signature is a non-empty signature when all nodes and edges in the graph representation of design pattern are named; otherwise, a signature is an empty signature.

(2009), can potentially enhance reasoning tasks. Presentation ODPs propose naming conventions for ontology elements to enhance usability and readability of ontology from a human perspective. Lexico-syntactic ODPs are patterns of linguistic structures that are automatically translatable into corresponding ontology elements.

Existing ODPs are systematised in NeOn (2012), Manchester (2009), and W3C (2005) libraries. Those existing ODPs are not necessarily adaptable to the requirements of domain-specific semantic platforms and tools, such is the MANU-SQUARE platform. Reusability of existing ODPs is always a good engineering practice, but specific tools and platform may have their own requirements, and thus, may require a platform-specific ODPs.

All-in-all, ODPs are very important for the ontology development process. ODP-guided ontology development leads to the uniformity and consistency in the design of ontology, in its architectural, presentational and computational aspects.

3.4 Ontology Development Tools

An ontology development tool, or simply ontology editor, is a software tool for the creation and manipulation of ontologies. Many different ontology development tools have emerged since the Semantic Web was born. Some of them stayed at the proof-of-concept level, while other one became widely-adopted and mature tools, with commercial and free licences. Ontology editors differ in many criteria: supported language, development technology, architecture, expressiveness, complexity, scalability, reasoning and querying plugins, maturity, licence, collaboration, interoperability, etc. A detailed comparison of the ontology editors has not been performed in Task 2.2 as there are already existing papers that provide the comparison by different criterion (e.g. Alatrish (2013), Seongwook and McLeod (2006), Sunitha and Babu (2013), Slimani (2015)). Below, a short overview of some of existing RDFS/OWL editors is given.

Apollo (The Open University, http://apollo.open.ac.uk/index.html) is a knowledge modelling application based around the basic primitives, such as classes, instances, functions, relations etc. Internal model is built as a frame system according to the internal model of the OKBC protocol. Apollo does a full consistency check while editing. Ontologies can be exported into RDF, XML, Meta and OCML formats. It is a desktop application written in Java, licence-free.

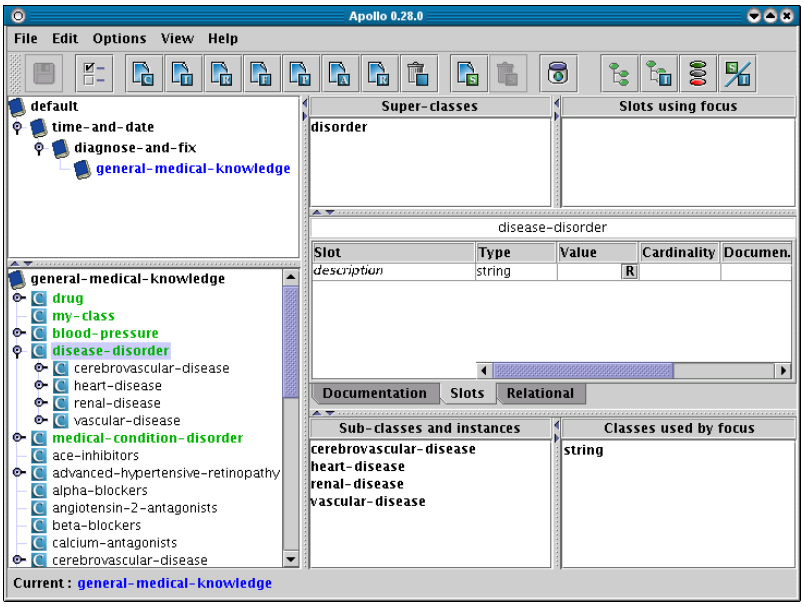


Figure 7 Apollo ontology editor

OntoStudio (http://www.semafora-systems.com/en/products/ontostudio/) is the commercial modelling environment for the creation and maintenance of ontologies. Among the most important features are the mapping tool, the graphic rule editor and the integrated test environment. With OntoStudio, several editors can provide and extend ontologies at the same time by using the OntoBroker Collaboration server. It is based on client/server architecture, where ontologies are managed in a central server and various clients can access and modify these ontologies. The internal representation

data model can be exported to DAML+OIL, F-Logic, RDF(S), OWL. Additionally, ontologies can be exported to relational databases via JDBC (Alatrish 2013). There are desktop and web versions of OntoStudio.

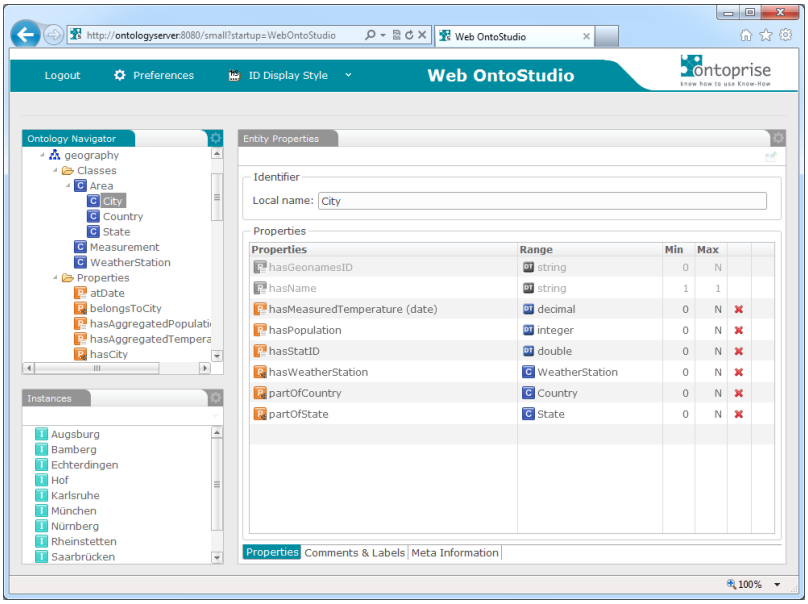


Figure 8 OntoStudio web ontology editor

Protégé (Stanford University School of Medicine, http://protege.stanford.edu/) is a free, open-source ontology editor that provides a suite of features to construct domain models and knowledge-bases. Supports the creation, visualization and manipulation of OWL ontologies. It can be extended by a plug-in architecture and Java-based application programming interface (API). Protégé allows the definition of classes, class hierarchies, property restrictions, and the relationships between classes and the properties of these relationships (T-Box). It allows declaration of instances of classes (A-Box) There are desktop and web versions of Protégé. The Web Protégé provides collaborative ontology development environment. Protégé offers DL reasoners for ontology consistency checks.

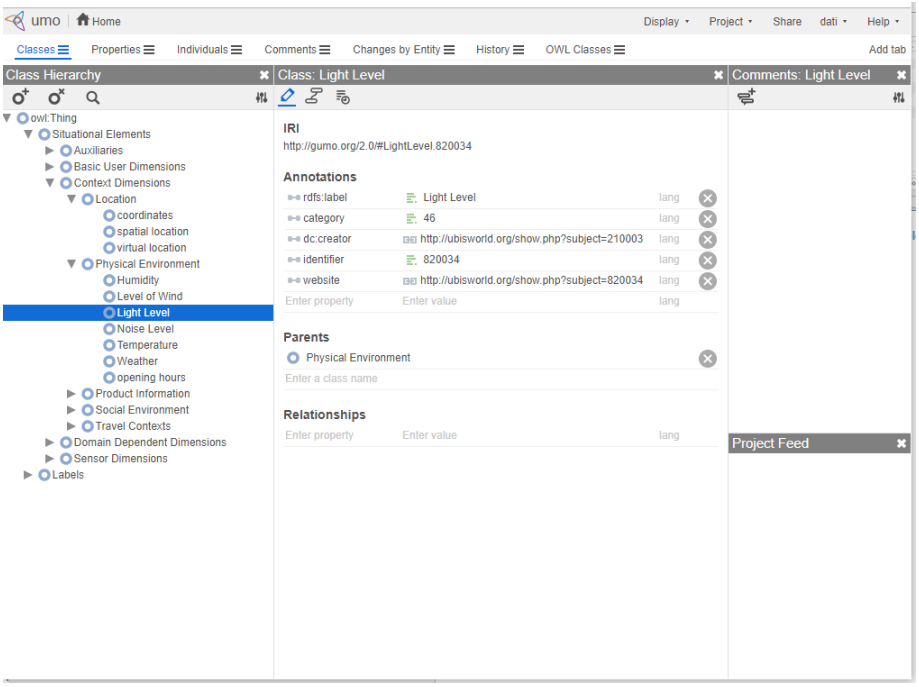


Figure 9 WebProtege ontology editor

SWOOP (https://github.com/ronwalf/swoop) is a hypermedia-based Featherweight OWL Ontology Editor, open-source, Web-based. It has formerly been maintained at the University of Maryland only, but then was jointly developed together with Clark & Parsia, IBM Watson Research and the University of Manchester. It contains OWL validation and offers various OWL presentation syntax views. It has reasoning support (RDFS-like and DL). Swoop is not maintained any more (it is an obsolete product).

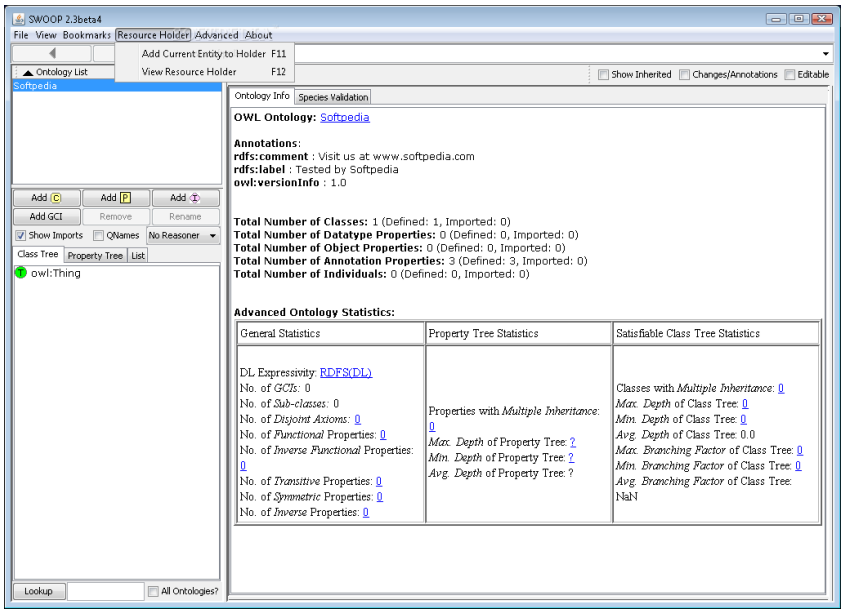


Figure 10 SWOOP ontology editor

NeOn Editor is a part of NeOn Toolkit that is the ontology engineering environment developed as part of the NeOn Project EU FP6 IST-2005-027595 (http://neon-project.org). It is an open source. The toolkit is based on the Eclipse platform and provides an extensive set of plug-ins covering a variety of ontology engineering activities, including annotation and documentation, development, ontology evaluation, ontology matching, reasoning and inference, reuse. It is a desktop application.

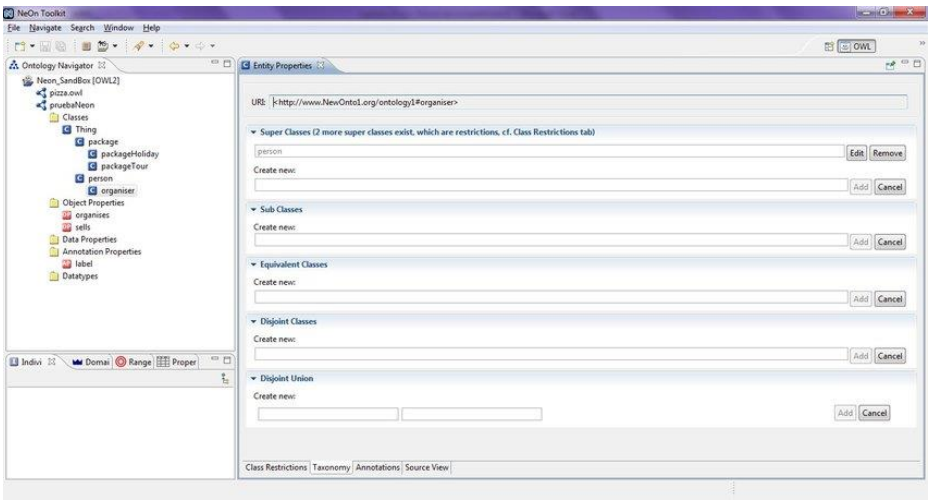


Figure 11 NeOn ontology editor

Vitro is a general-purpose web-based ontology and instance editor with customizable public browsing. Vitro is a Java web application that runs in a Tomcat servlet container. Vitro can be used to create or load ontologies in OWL format, edit instances and relationships, search data with Apache Solr. It has been demonstrated at the 2011 International Conference on Biomedical Ontologies. Source code is available on https://github.com/vivo-project/Vitro.

OWLGrEd is free UML style graphical editor for OWL ontologies. It has additional features for graphical ontology exploration and development, including interoperability with Protege. Available at: http://owlgred.lumii.lv/. There are desktop and web versions of OWLGrEd.

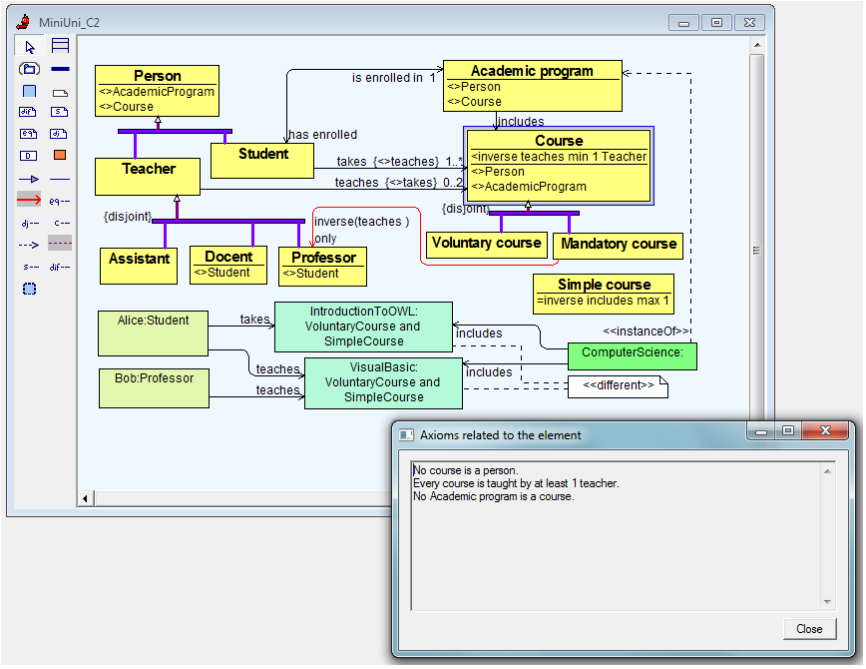


Figure 12 OWLGrEd ontology editor

WebVOWL (http://vowl.visualdataweb.org/webvowl.html#editor) is a web application for the interactive visualization of ontologies. It implements the Visual Notation for OWL Ontologies (VOWL) by providing graphical depictions for elements of the Web Ontology Language (OWL) that are combined to a force-directed graph layout representing the ontology. Interaction techniques allow to explore the ontology and to customize the visualization. The VOWL visualizations are automatically generated from JSON files into which the ontologies need to be converted. WebVOWL is able to visualize most language constructs of OWL 2 but not all of them yet (and also not all combinations). For instance, complex data types and some instance level constructs are not supported by WebVOWL at the moment.

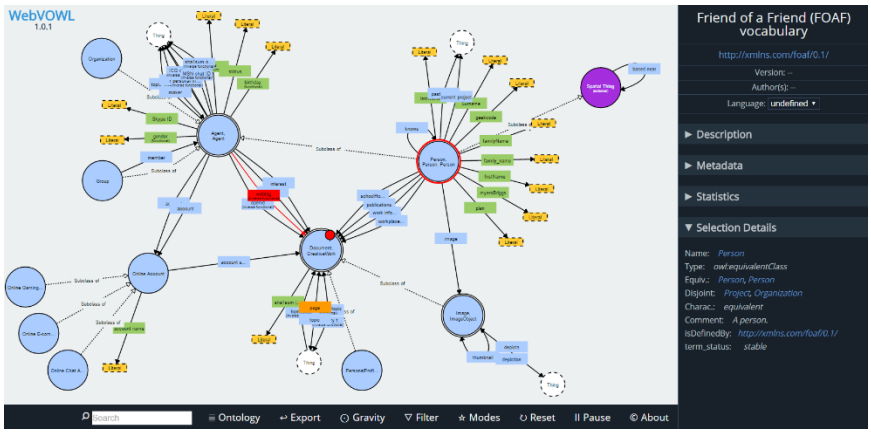


Figure 13 WebVOWL browser

TopBraid Composer (TopQuadrant) comes in two editions. Standard Edition (SE) includes all fundamental features for ontology editing plus graphical viewers, import facilities, advanced refactoring support. Maestro Edition (ME) provides SE features plus additional features. It is based on the Eclipse platform and the Jena API. It is a complete editor for RDF(S) and OWL models, as well as a platform for other RDF-based components and services. TopBraid Composer (FE) can load and save any OWL2 file in formats such as RDF/XML or Turtle. As pointed in Alatrish (2013), TopBraid Composer

supports various reasoning and consistency checking mechanisms. Consistency checking and debugging is supported by built-in OWL inference engine, SPARQL query engine and Rules engine. OWL description logic is supported via a range of built-in OWL DL engines such as OWLIM, Jena and Pellet. It supports SPARQL inference Notation (SPIN).

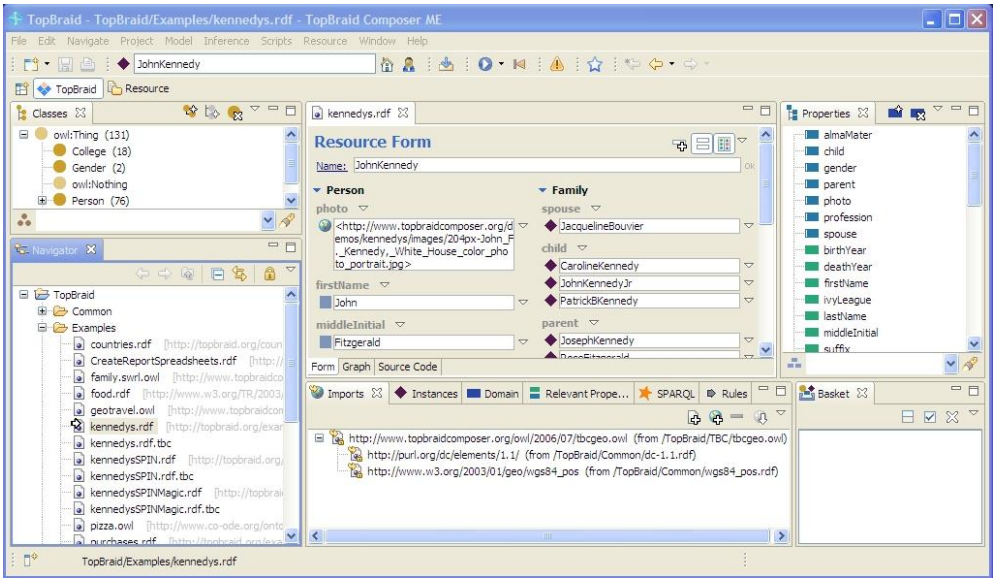


Figure 14 TopBraid Composer

Besides the above described editors, there are also other ones e.g. Knoodl, Semantic Turkey, WebODE, FluentEditor (uses Controlled Natural Language) (see https://www.w3.org/wiki/Ontology_editors). Figure 15, taken from Sunitha and Babu (2013), provides a comparison table of the ontology development tools.

International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June-2013 1751
ISSN 2229-5518

TABLE 1
COMPARISON OF ONTOLOGY DEVELOPMENT TOOLS

Tool / Features	Ontolingua Server	OntoSaurus	OilEd	WebOnto	Protégé	SWOOP	TopBarid Composer	WebODE	OntoEdit	Neon Toolkit
Availability	Free	Free & Open	Free & Open	Free	Free	Free & Open	Commercial	Free	Free	Free & Open
Versioning	No	No	No	Y/N	Y/N	Yes	Y/N	No	Y/N	Yes
Collaborative	Yes	Yes	No	Yes	Yes (Collaborative Protégé)	Yes	Yes	Yes	No	Yes
Graphical Class/Property taxonomy	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Back up management	No	No	No	Yes	No	Y/N	Y/N	Yes	No	Yes
Support growth of large ontologies	Yes	Y/N	No	Y/N	Yes	Yes	Y/N	Yes	Y/N	Yes
Querying	No	No	No	Y/N	Yes	No	Yes	No	Y/N	Yes
User Interface	No	Y/N	Yes	Y/N	Yes	Yes	Yes	Yes	Y/N	Yes
Consistency check	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
OWL Editor	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Extensibility	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Ontology Libraries	Yes	No	Yes	Yes	Y/N	Y/N	Y/N	No	No	Yes
Architecture	Client/Server	Client/Server	Standalone	Client/Server	Standalone	Standalone	Client/Server	N-Tire	Standalone	Standalone
KR Paradigam of Knowledge model	Frames+, FOL	DL	DL	Frames+, FOL	Frames+, FOL+, Meta classes	DL	DL	Frames+, FOL	Frames+, FOL	DL
Import	Ontolingua, DAML+OIL, CLIPS	LOOM, IDL, KIF, C++	RDF(S), DAML+OIL	OMCL	RDF(S), OWL	RDF(S), OWL	RDBMS, OWL, RDF(S)	RDF(S), DAML+OIL, OWL	RDF(S), DAML+OIL	RDF(S), OWL
Export	Ontolingua, DAML+OIL, CLIPS	LOOM, IDL, KIF, C++	RDF(S), DAML+OIL, OWL	OMCL, Ontolingua, RDF(s), OIL	RDF(S), OWL, CLIPS	RDF(S), OWL	OWL, RDF(S), XML	RDF(S), DAML+OIL, OWL, CLIPS	RDF(S), DAML+OIL, OWL	RDF(S), OWL
Storage	Files	Files	Files	Files	Files, DBMS(JDBC)	Files	Files	DBMS(JDBC)	Files	Files
Reasoner	JTP, Prolog, CML, Epikit	PowerLoom, Stella	FaCT	-	Pellet	Pellet	Pellet	Prolog	OntoBroker	Pellet2, Hermit, OntoBroker
Merging	Chimaera	None	None	None	Prompt, OWLDiff	Yes	Y/N	ODE Merge	Yes	Yes
Debug/Repair	No	No	Very Little	No	Very Little	Yes	No	No	No	Yes
Built-in Inference	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Y/N	Yes
Implemented in	Lisp	Lisp	Java	Lisp	Java	Java	Java	Java	Java	Java Eclipse

Figure 15 Comparison of Ontology editors (source: Sunitha and Babu, 2013)

The described editors allow an "ad-hoc" ontology development which is an ontology development process that does not strictly enforce a particular ontology development methodology and ontology design rules for application-depended ontologies. In the MANU-SQUARE case, domain-specific ontologies should adhere to predefined design rules, principles and guidelines, in order to preserve the uniform ontology design, thus, to preserve the matchmaking functionalities. Otherwise, if the ontology development process is uncontrolled and not guided by the tool, the ontology may become unusable for already established matchmaking queries and other related tasks. For example, a design rule might be: all *domain-specific concepts must be subclasses of core concepts*, or (Rector, 2005)'s recommendation for descriptive features - *descriptive features must be captured as disjoint classes which exhaustively partition the parent class representing the feature*. Therefore, the MANU-SQUARE platform needs an ontology editor that should be able to enforce established development methodology and design rules.

4 MANU-SQUARE DOMAIN-SPECIFIC ONTOLOGY DEVELOPMENT

The consistent and quality development of the MANU-SQUARE domain-specific ontologies is a challenge because the ecosystem deals with a large and evolving manufacturing domain. It is unfeasible within the scope of the MANU-SQUARE project to capture and formalize all existing manufacturing knowledge of different manufacturing sectors into the ecosystem's knowledge base. Thus, the initial versions of MANU-SQUARE domain-specific ontologies have been captured and scoped primarily to support execution two industrial demonstration of the platform:

- unused capability discovery;
- new products development and production in AGV and textile-cosmetics sector.

Being initial, it is expected that the ontologies will evolve further by adding in, by platform managers and domain experts, additional concepts related to the target sectors. The development of evolving domain-specific ontologies in MANU-SQUARE is an iterative-incremental and should follow a methodological and structured process that leads to the uniformity and consistency in the design of ontologies.

4.1 Methodology

To develop initial versions of MANU-SQUARE domain-specific ontologies for the use-cases of unused manufacturing capability discovery, and new products development/production in AGV and textile-cosmetics sector, the following steps have been taken:

1. Requirements specification and analysis. This phase involves the requirements analysis and has these activities:
    a. Determining the domain of interest and the early scope of ontology
    b. Defining the purpose or motivating scenario (use-cases)
    c. Identifying the competency questions within use-cases
    d. Competency questions-based interview with domain experts (knowledge elicitation)
2. Design and conceptualization
    a. Identification of relevant terms (domain lexicon) and the glossary building (based on collected Competency questions and interviews with domain experts)
    b. Categorisation of concepts
    c. Refinement of the concepts and their relations
3. Implementation
    a. Formalization/coding in OWL/RDFS language, according to recommended design principles
        i. Create classes
        ii. Create semantic relationships between classes (taxonomy development)
        iii. Create properties of classes, if any
        iv. Create instances of classes, if any
    b. Extension of MANU-SQUARE Core concepts, if needed
    c. Integration with existing ontologies of the ecosystem, including with MANU-SQUARE Core concepts ontology
    d. Integration with external ontological resources using seeAlso relationship
4. Syntax verification and consistency check (using reasoners)

5. Evaluation (with SPARQL queries that check if ontology is able to answer the competency questions)

MANU-SQUARE domain-specific ontology development methodology (MODE, in short) builds on methodologies introduced in § 3.2.

MODE follows the UPON workflows - Requirements, Analysis, Design, Implementation, and Test, but also has some of the steps from 101-Method. However, it **does not strictly** adhere to any of introduced approaches as MANU-SQUARE platform has specific requirements for ontology development. These specifics are: (1) domain-specific ontologies must adhere to the MANU-SQUARE Core concepts model must be specialization or extensions of the core concepts; (2) domain-specific ontologies may constantly evolve within existing and across new vertical sectors.

Methodologies such as UPON and METHONTOLOGY emerged from the projects that needed the finalized (completed ontologies), which do not change or evolve into vertical domains over the time. However, MANU-SQUARE's requirements are significantly different, thus, the development methodology is different. The MANU-SQUARE ontologies may evolve across manufacturing sectors and new domain specific concepts may emerge. The introduction of new concepts into existing ontologies must not be performed in an "ad-hoc" manner, but should follow the methodology, notably in implementation steps where ontology design recommendations must are applied.

Please note that steps 2-a and 2-b may can include semi-automated means for domain terms extraction and their categorization into concepts. However, the ontology implementation is always a manual process, but it can supported by an ontology editor that provides graphical user interface to ease the implementation tasks for domain experts. (Domain experts in most cases do not have knowledge of ontology languages and rule languages).

The MODE is illustrated in Figure 16.

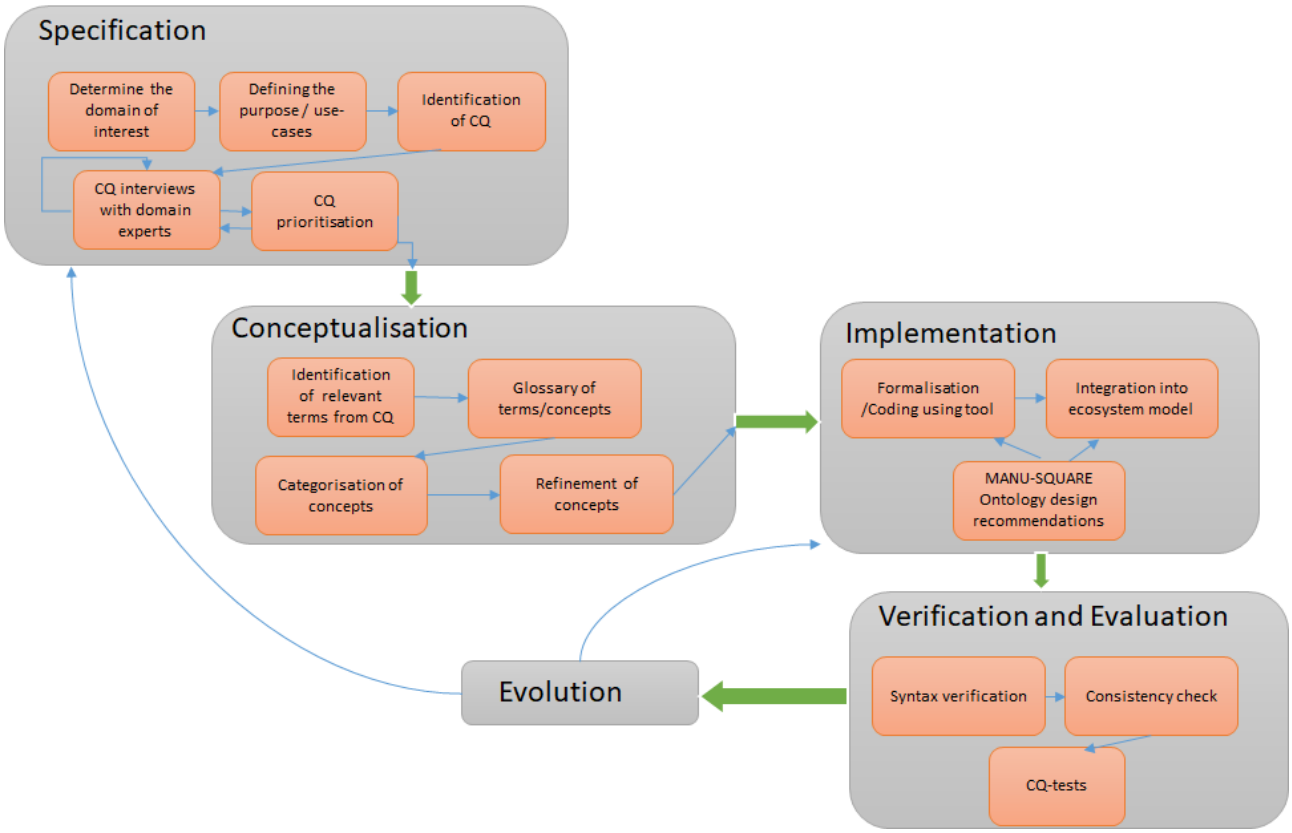


Figure 16 MANU-SQUARE domain-specific ontology methodology

The MANU-SQUARE ontology-editor tool, enforces the application of MANU-SQUARE ontology design recommendations. Figure 17 illustrates **one of the design patterns,** CapabilityType modelling pattern, and a corresponding structural template of graphical user interface.

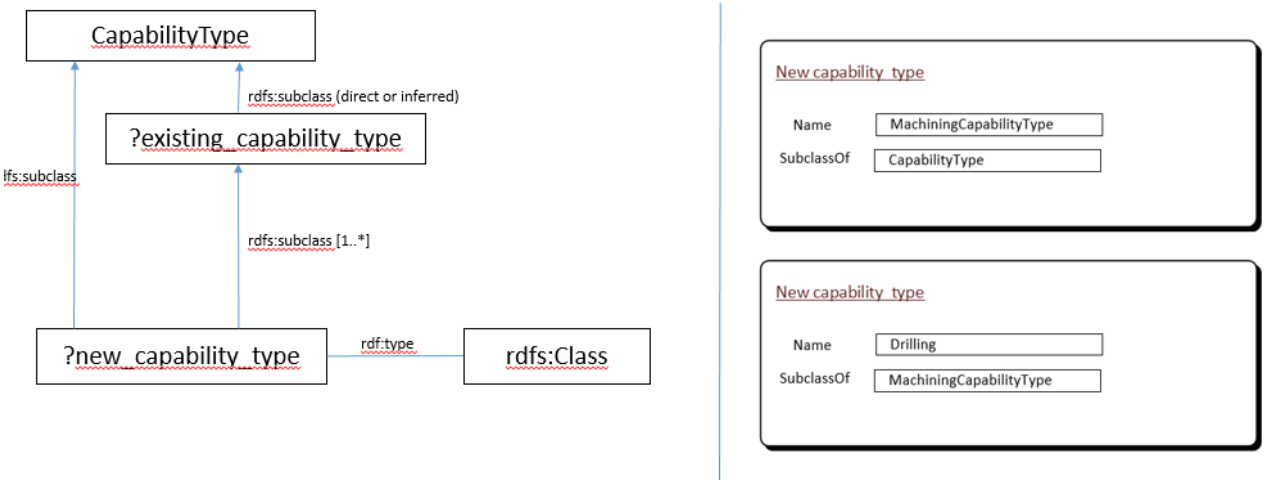


Figure 17 Illustration of MANU-SQUARE design pattern for CapabilityType taxonomy modelling

4.2 Applying the methodology

As described in D1.3 (Business processes and early validation scenario), the domains that a MANU-SQUARE MVP will be demonstrated in include:

- Machining industry that include scenario of a new product development (production of an AGV solution) and a market coverage expansion (expanding to the retrofitting market) for the pilot company (JPM).
- Textile & cosmetics sector embedded to each other, with a scenario where one of the pilot companies (TRUDEL) provides its by-product as a raw material (Sericin) to the other pilot company (I-COTTON) for the development of new product (wet-wipes).

The development of domain-specific ontologies was done by following the MODE methodology. Domain of interest and motivation scenarios (use-cases) were identified in task Task 1.3, while MANU-SQUARE Core Factory model was produced in Task 2.1. These were the staring points. Then, CQ technique (see § 3.2) was performed with the group of domain experts from JPM, I-COTTON, TRUDEL, I-HUB, CSEM companies. After the CQ were collected and domain knowledge elicited with the questionnaires (see Appendix A and B), we held the interviews with the domain expert to identify the scope of ontologies and the key CQ-related terms in the respective domains. The key terms are then conceptualized and organized into taxonomies of equipment and human capability types, item types, process types, attribute types, resource types. During the implementation phase, the domain-specific concepts are linked to the MANU-SQUARE Core Model. In other words, the MANU-SQUARE Core Model is extended with specific concepts and their taxonomies in the production engineering (AGV), retrofitting sector, and textile-cosmetics sectors. And this extension forms the domain-specific ontologies. Encoding is done using RDFS/OWL and during the implementation design principles are followed to achieve uniform and consistent design throughout the ontology.

As illustrated in Figure 18, the main extension points of MANU-SQUARE Core Model are the root concepts of different taxonomies. These are Process Types, Item Types, KPI Types, Attribute Types, Capability Types, and Energy Types. The taxonomical organisation enables the matchmaking using the subsumption computation.

In Figure 18, only ItemType taxonomy is shown with more levels just for illustration purpose; § 5 elaborates all the taxonomies in more details.

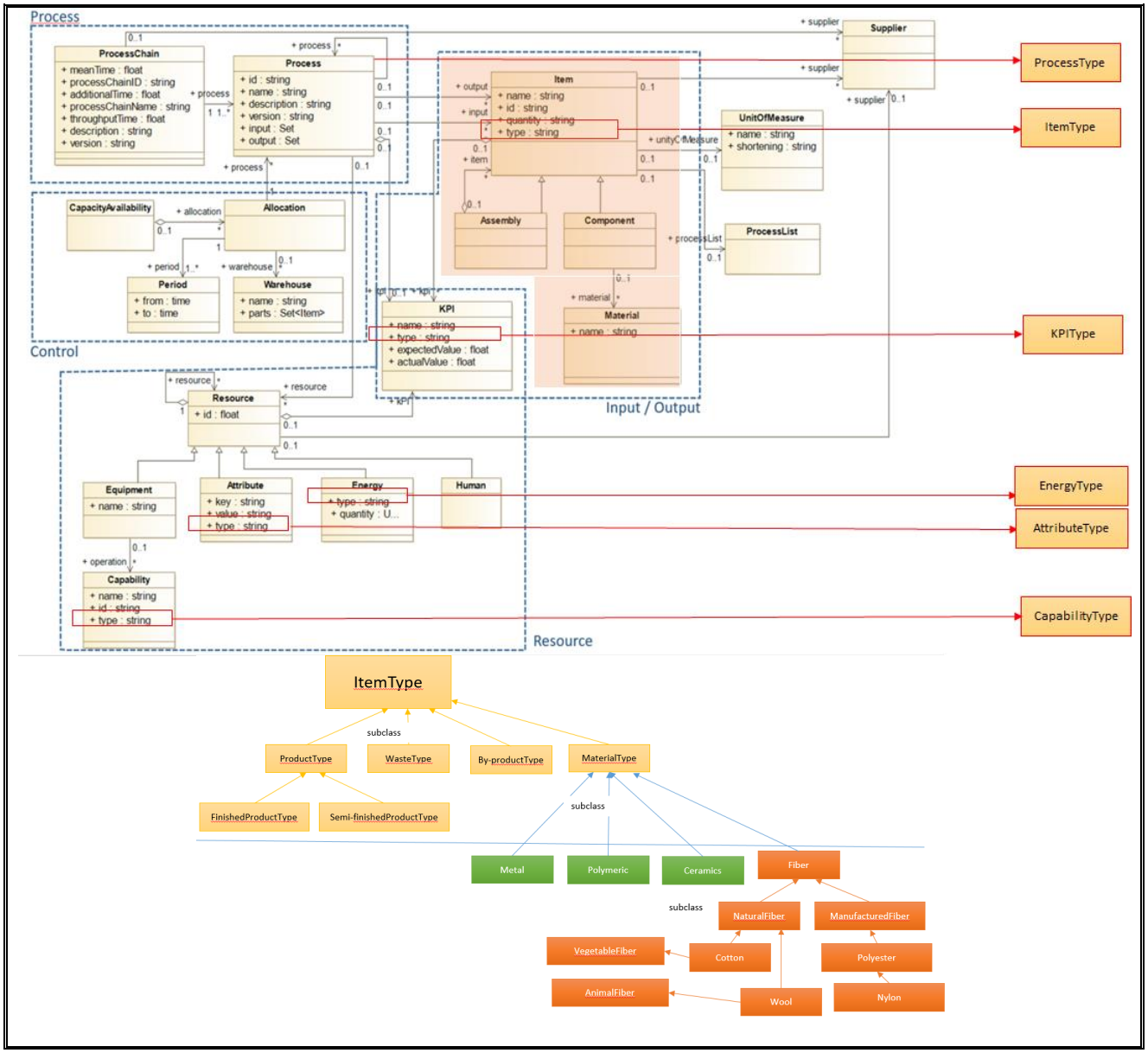


Figure 18 Extension of MANU-SQUARE Core Concepts

In addition to domain-specific concepts are their taxonomies, MANU-SQUARE Core Factory model is extended with some new core concepts and relationships. These are:

- **ProcessInput** concet – to directly classify items that are input to the process
- **ProcessOutput** concept - to directly classify items that are output of the process
- **Industry** concept - to allow categorisation of suppliers by industry sectors then belong to or they serve
    - **servesIndustry** property between Supplier and Industry – to specify for which industries manufacturing supplier usually works with (has specialization for certain industries e.g. automotive, military, etc)
    - **belongsToIndustry** property between Supplier and Industry – to specify in which industrial sector manufacturing supplier belongs to
- **hasAttribute** relationship between Process and Attributes, and between Item and Attribute. The production processes as well as items can have their attributes that characterize them (e.g. Item can have measured attributes such as length, width, height, etc.)

In order to group domain specific concepts into their respective domains, the domain-specific ontology has classes that represent the sector. For example, **AGVSectorClass** groups concepts and taxonomies that belong to the AGV sector

only. Similarly, **TCSectorClass** groups concepts and taxonomies from the textile-cosmetics sector. **AnySectorClass** categorises taxonomies and concepts that are more specific than the core concepts, but are not sector-specific. For example, classification of industries, or classification of certifications are AnySectorClass category.

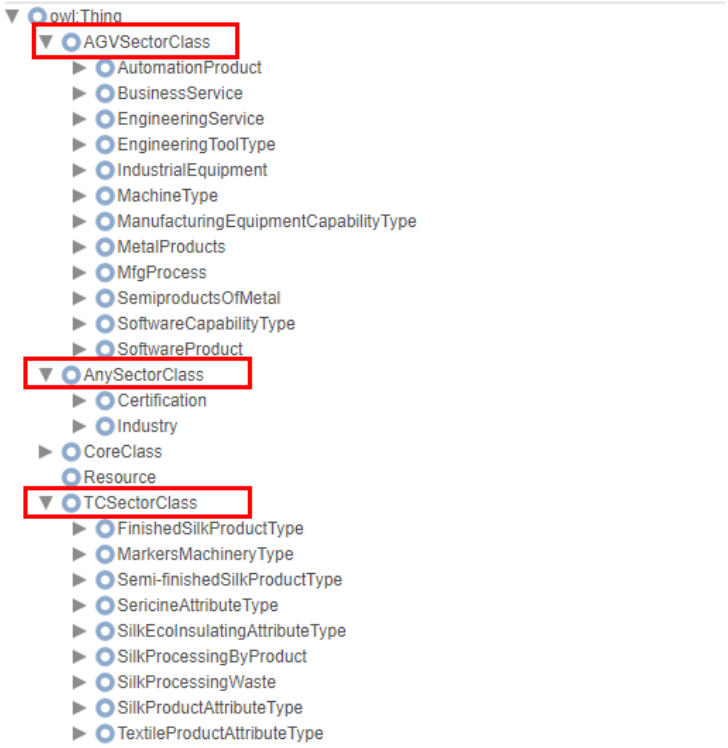


Figure 19 Sector-grouping classes

Further, in order to avoid unnecessary explicit statements in the suppliers/factories descriptions, a first set of core inference rules have been defined. These are,

- rule1: (?process :hasInput ?item) (?item rdf:type :Item) -> (?item rdf:type :ProcessInput]
- rule2: (?process :hasOutput ?item) (?item rdf:type :Item) -> (?item rdf:type :ProcessOutput]
- rule3: (?item rdf:type :Item), noValue(?item :hasItems) -> (?item rdf:type :Component]
- rule4: (?item rdf:type :Item), (?item :hasItems ?item-x) -> (?item rdf:type :Assembly]
- rule5: (?material rdf:type :Material), (?material :hasItems ?material-x) -> (?material rdf:type :CompositeMaterial]

Hence, instead of explicitly asserting that a certain item is ProcessInput or ProcessOutput, the system can execute the rule 1 and rule 2, to automatically derive this classification based on the rest of the description of the resource. Similarly, for Assembly and CompositeMaterial classifications, there is no need to explicitly provide the types in a description of items that are assembly or composite, as the classification can be done by the rules 4 and 5.

4.3 Reusability for new MANU-SQUARE sectors

The ontology development methodology introduced in § 4.1 can be applied for new vertical sectors that potentially will be integrated in MANU-SQUARE ecosystem. The development may involve domain experts or may be based on semi-automated knowledge extraction from textual resources. In either case, the platform manager or domain expert would be responsible to properly capture new concepts and taxonomies using the MODE ontology editor, and to integrate new ontologies into the ecosystem.

5 MANU-SQUARE DOMAIN-SPECIFIC ONTOLOGIES

This section presents domain-specific ontologies developed for AGV sector and textile-cosmetics-silk sector. There are about 1012 concepts in the integrated domain-specific ontology, counted together with concepts of MANU-SQUARE Core ontology.

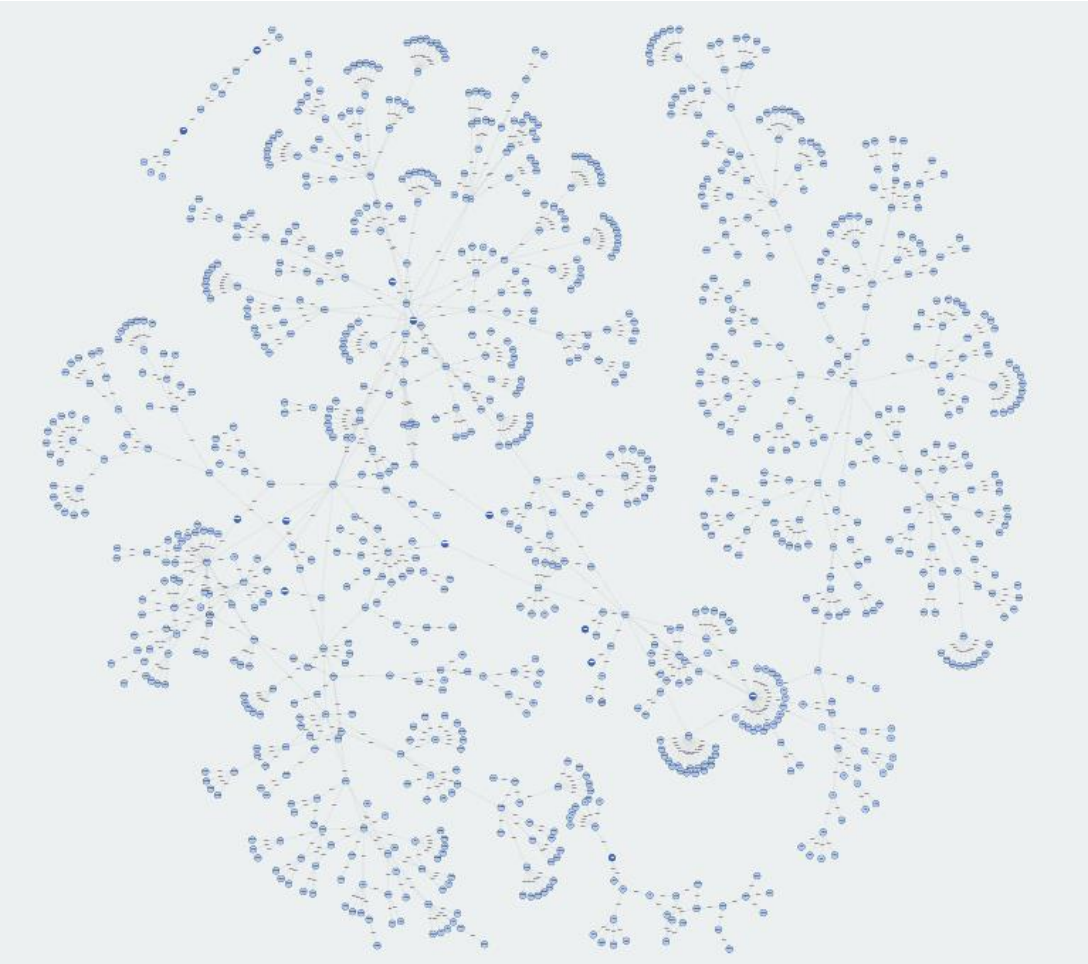


Figure 20 MANU-SQUARE domain-specific ontologies as seen in WebVOWL tool

The following sections (§ 5.1 and § 5.2) present the developed domain-specific extensions of MANU-SQUARE core concepts. § 5.1 reports the extensions for AGV and machining sector, while § 5.2 reports the extensions for Textile-Cosmetics-Silk Processing/Products sector. The presented overview of these extensions (taxonomies) in the following sections is not an exhaustive overview of all the domain-specific concepts so far captured, as that would be of inappropriate size (text length) for this document purpose.[6]

Hence, for each domain section, tables reports some excerpt of the respective domain ontologies. In each table, the picture on the left reports just a specific part of the taxonomy, from an ontology authoring tool, while pictures on the right give a more complete understanding of the actual graph of the ontology (from the leaf to the roots) for a selected term.

[6] OWL file of domain-specific extensions of MANU-SQUARE core ontology is available here:
https://github.com/vujasm/mode/tree/master/src/main/resources/ontologies/MANU-SQUARE-industrial.owl
while the OWL file of the core ontology is available here:
https://github.com/vujasm/mode/tree/master/src/main/resources/ontologies/MANU-SQUARE-core.owl

Machining domain ontology reuses several portions of a MSDL ontology (Ameri and Dutta 2006). These portions are: portions of taxonomy of manufacturing services, portions of taxonomy of machining equipment, and portions of taxonomy of products type.

5.1 Manufacturing/Engineering Sector concepts

5.1.1 ProcessTypes

Table 1 shows a part of the developed ProcessType taxonomy.

Taxonomy

- ProcessType
  - MfgProcess
    - AssemblyProcess
    - NonShapingProcess
    - ShapingProcess
      - AdditionProcess
        - RapidPrototyping
          - DepositionBasedPrototyping
          - LaserBasedPrototyping
          - Three-DPrinting
      - ConsolidationProcess
        - ChemicalJoining
        - MechanicalJoining
        - ThermalJoining
      - DeformationProcess
      - SolidificationProcess
      - SubtractionProcess
        - ChemicalSubtraction
        - MechanicalSubtraction
          - AbrasiveMachining
            - Burring
            - Deburring
            - Grinding
              - CNCGrinding
            - Honing
            - JetMachining
            - Lapping
            - UltrasonicMachining
          - MultiPointCutting
          - SinglePointCutting
        - ThermalSubtraction
    - TestingProcess

Graph visualisation

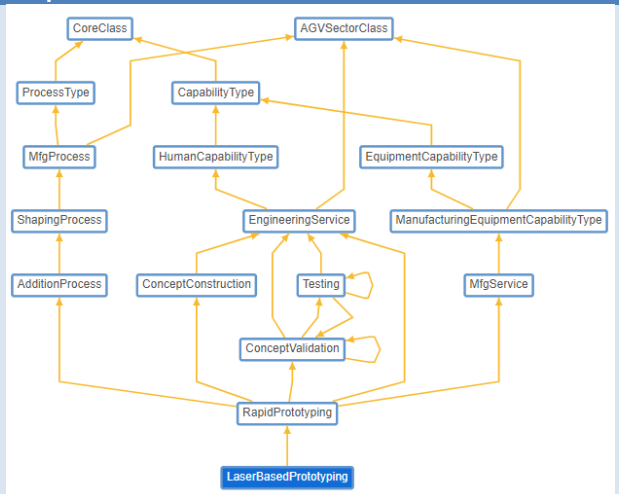


Table 1 Process Type taxonomy in AGV/Machining sector

5.1.2 ItemTypes

Table 2 shows a part of the developed Item Type taxonomy.

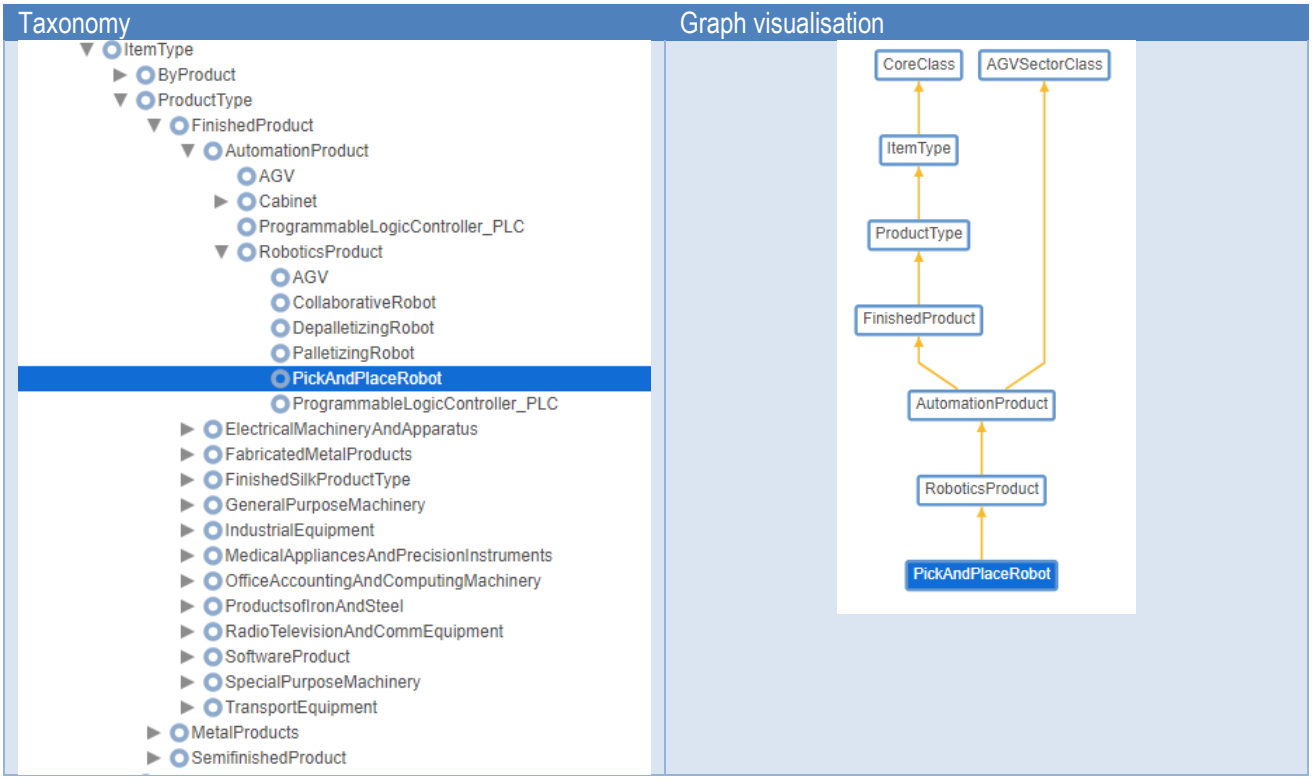


Table 2 Item Type taxonomy in AGV/Machining sector

5.1.3 Material Types

Table 3 shows a part of the developed Material Type taxonomy.

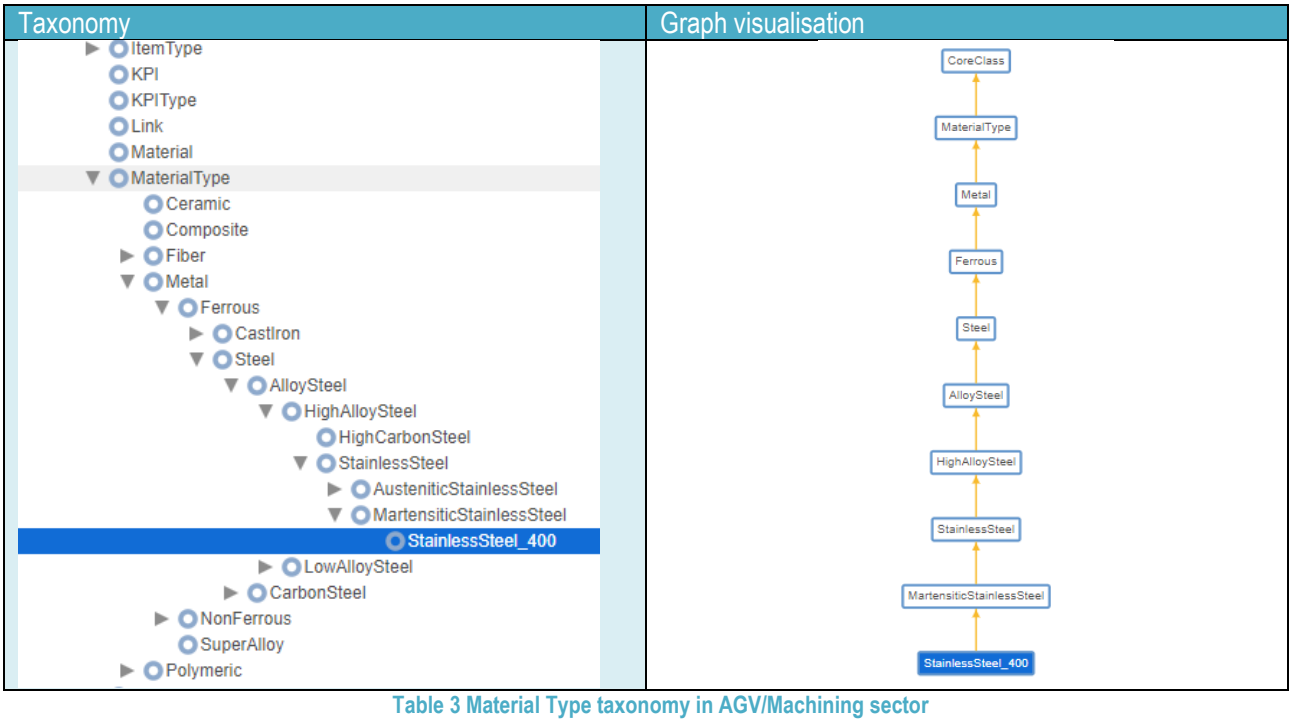


Table 3 Material Type taxonomy in AGV/Machining sector

5.1.4 Manufacturing Equipment Types

Table 4 shows a part of the developed Manufacturing Equipment Type taxonomy.

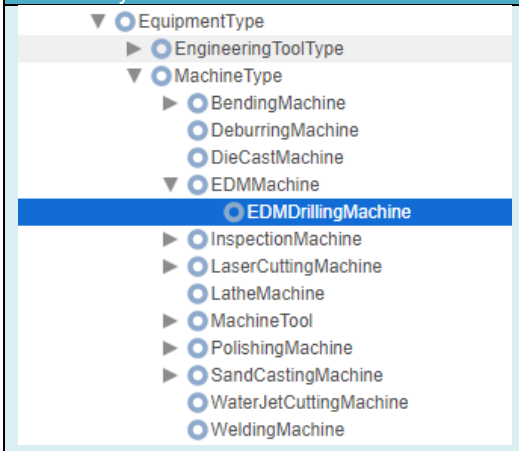
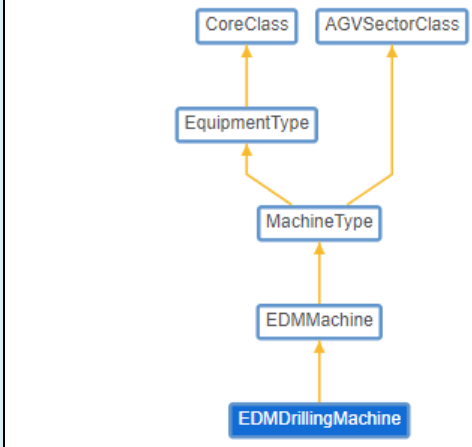
Taxonomy	Graph visualisation
	

Table 4 Equipment Type taxonomy AGV/Machining sector

5.1.5 Engineering Equipment Types

Table 5 shows a part of the developed Engineering Equipment Type taxonomy.

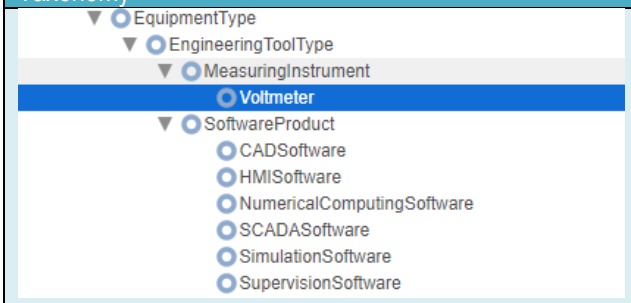
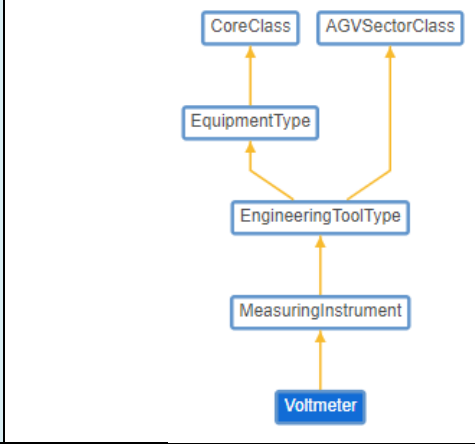
Taxonomy	Graph visualisation
	

Table 5 Engineering Equipment Type taxonomy in AGV/Machining sector

5.1.6 Manufacturing Equipment Capabilities

Table 6 shows a part of the developed Manufacturing Equipment Capabilities taxonomy.

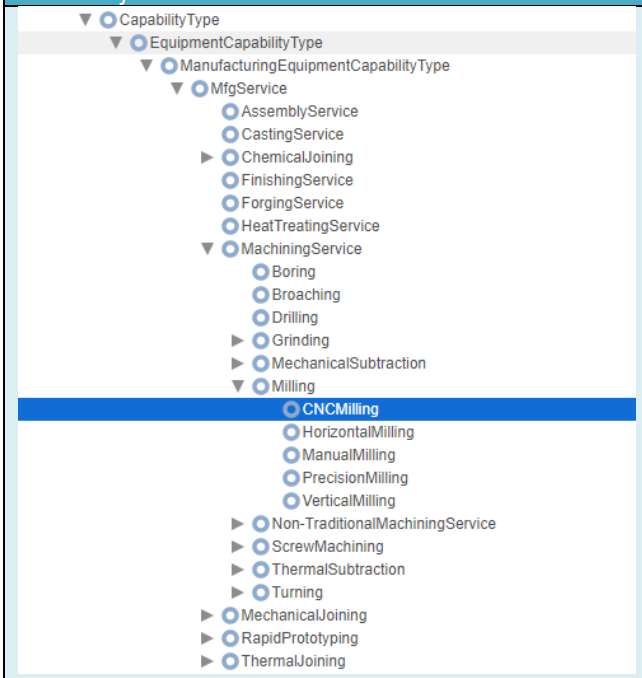
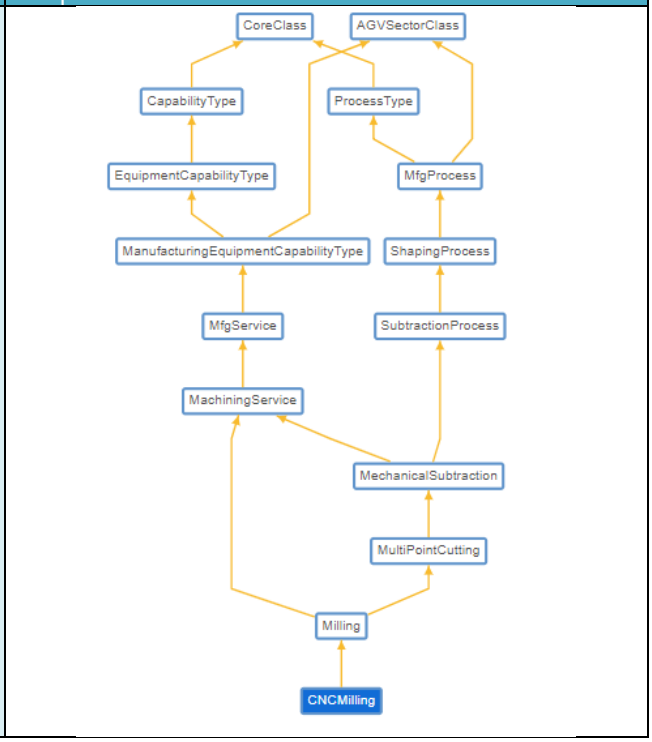
Taxonomy	Graph visualisation
	

Table 6 Manufacturing Equipment Capabilities taxonomy

5.1.7 Software Capabilities

Table 7 shows a part of the developed Software Capabilities taxonomy.

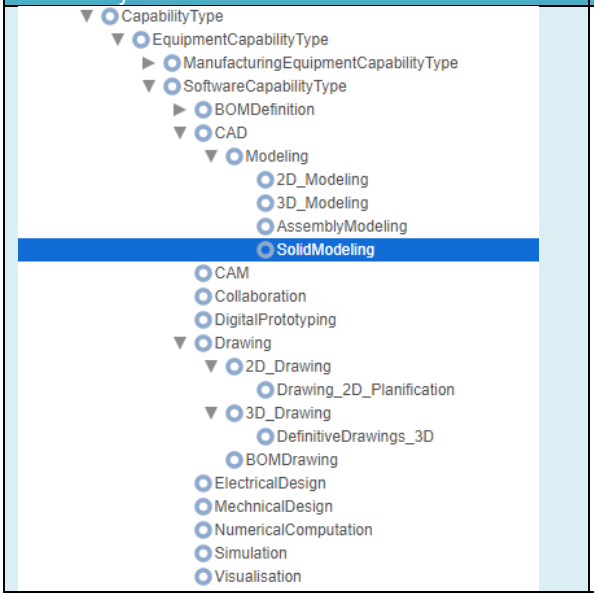
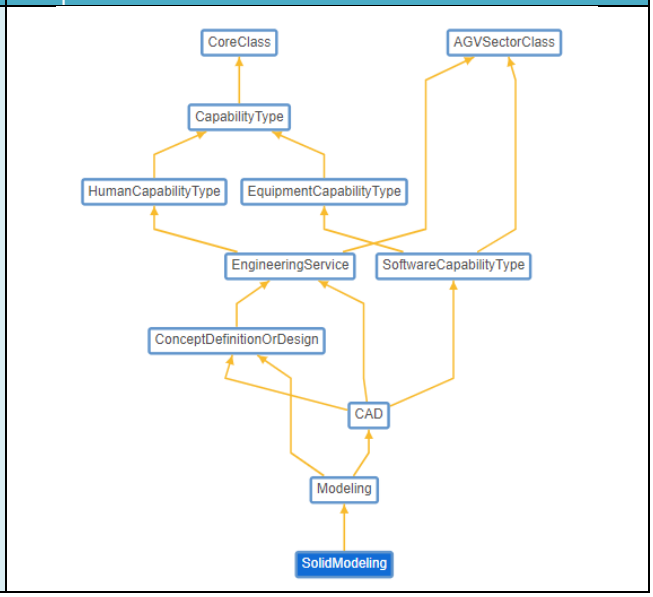
Taxonomy	Graph visualisation
	

Table 7 Software Capabilities taxonomy

5.1.8 Human Capabilities

Table 8 shows a part of the developed Human Capabilities taxonomy.

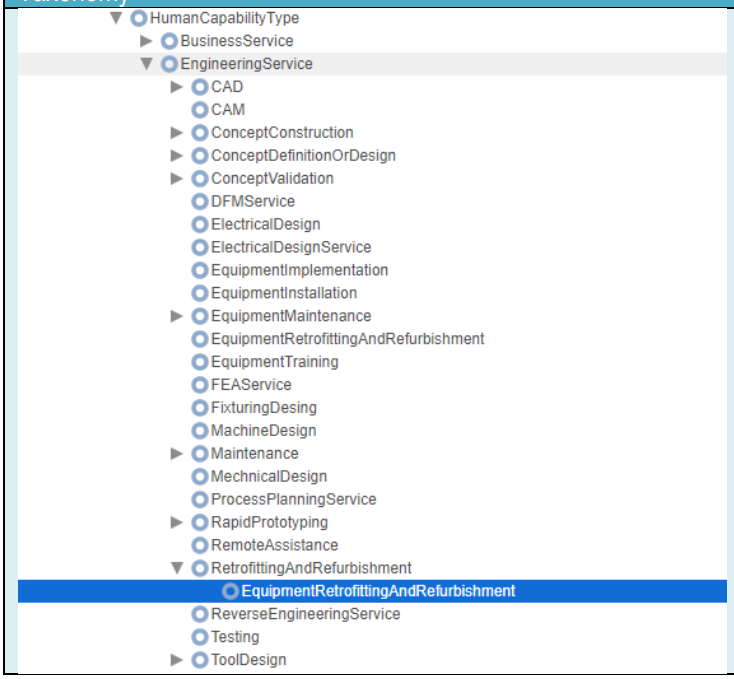
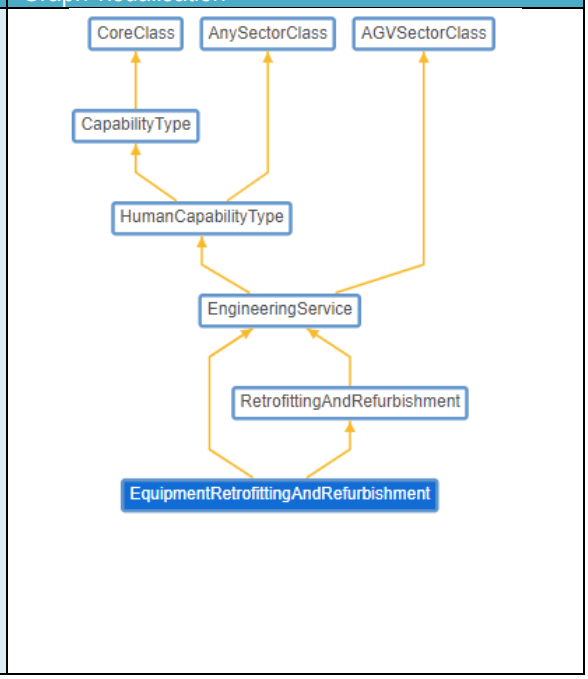
Taxonomy	Graph visualisation
	

Table 8 Human Capabilities taxonomy in AGV/Machining sector

5.1.9 Attribute Types

Table 9 shows a part of the developed Attribute Types taxonomy.

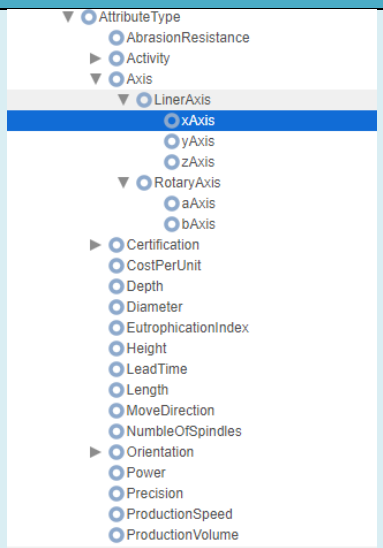
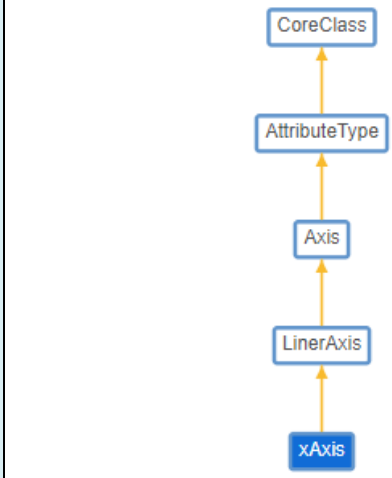
Taxonomy	Graph visualisation
	

Table 9 Attribute Type taxonomy in AGV/Machining sector

5.2 Textile-Cosmetic Sector concepts

5.2.1 ItemTypes

Table 10 shows a part of the developed Item Types taxonomy.

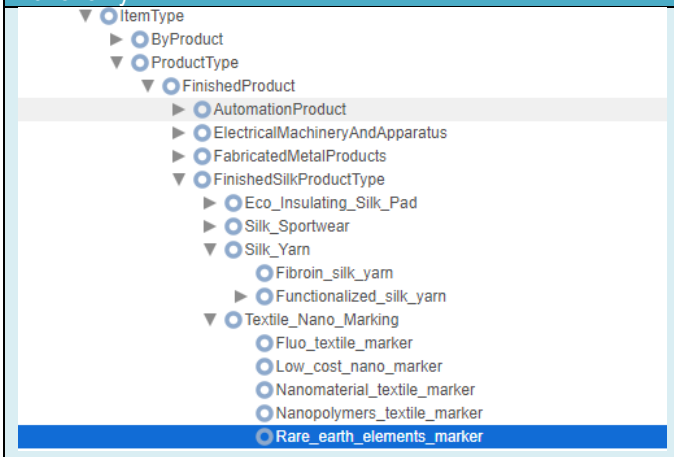
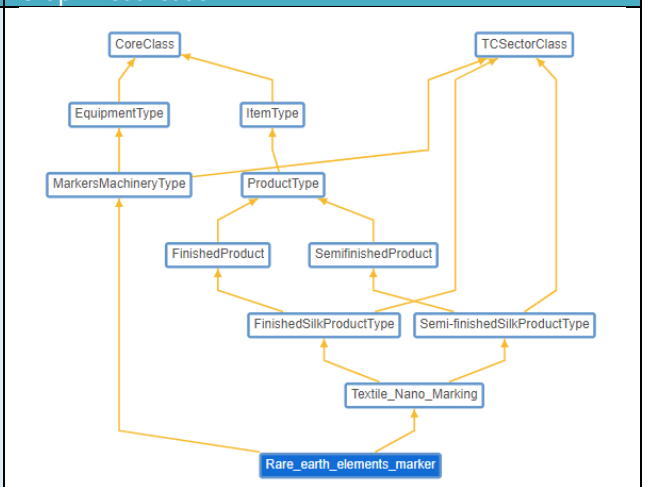
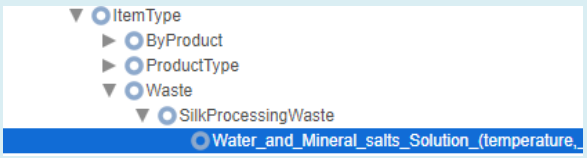
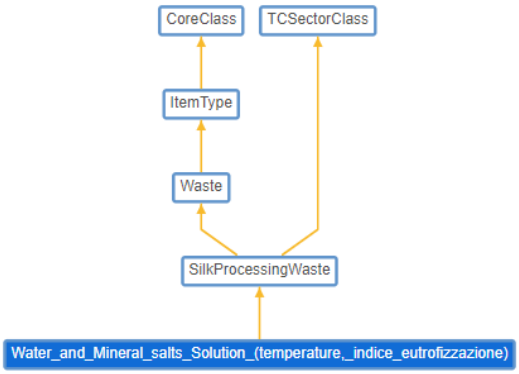
Taxonomy	Graph visualisation
	
	

Table 10 Item Type taxonomy in Textile-Cosmetics Sector

5.2.2 Material Types

Table 11 shows a part of the developed Material Types taxonomy.

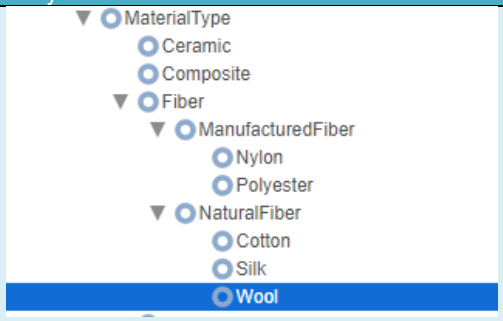
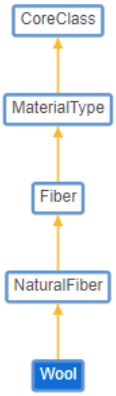
Taxonomy	Graph visualisation
	

Table 11 Material Types taxonomy in Textile-Cosmetics Sector

5.2.3 Markers Machinery Types

Table 12 shows a part of the developed Markers Machinery Types taxonomy.

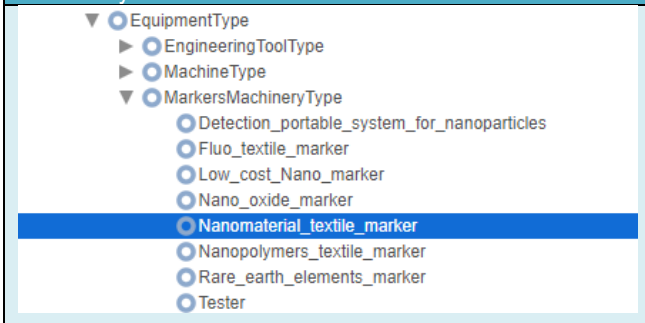
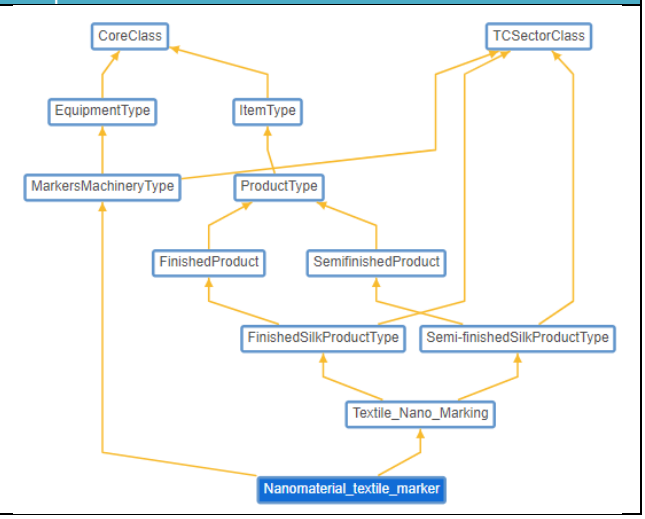
Taxonomy	Graph visualisation
	

Table 12 Markers Machinery Types taxonomy in Textile-Cosmetics Sector

5.2.4 Attribute Types

Table 13 shows a part of the Attribute Types taxonomy.

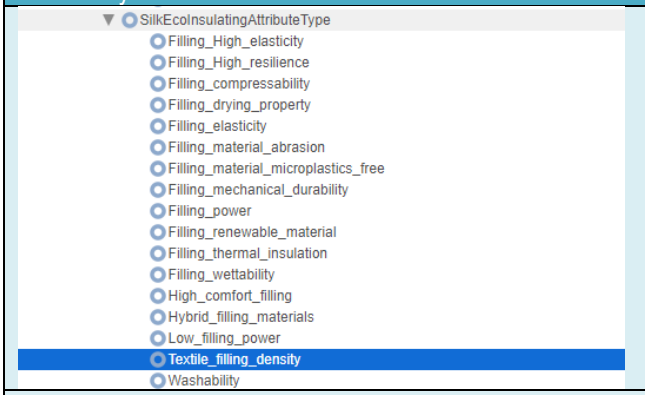
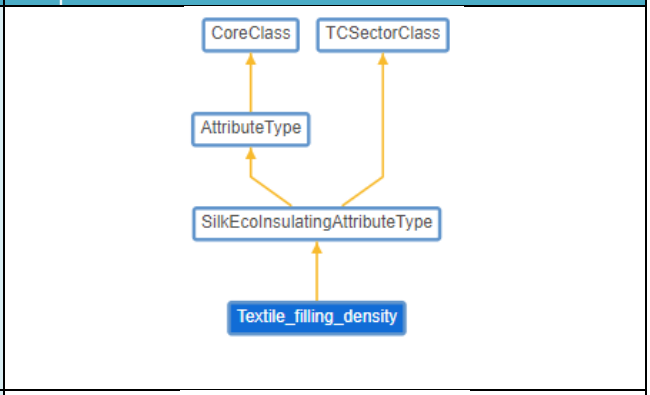
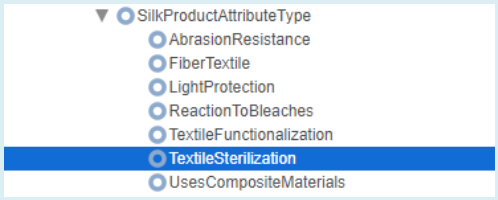
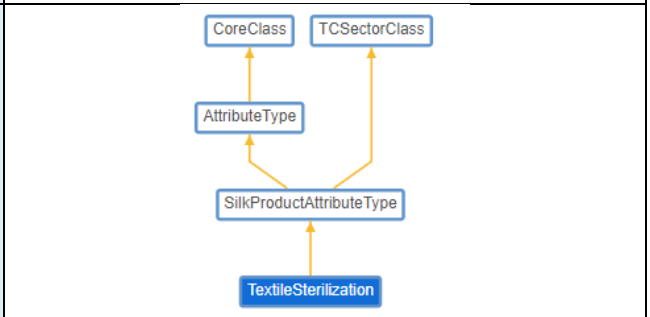
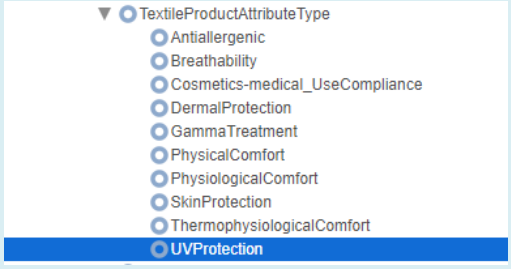
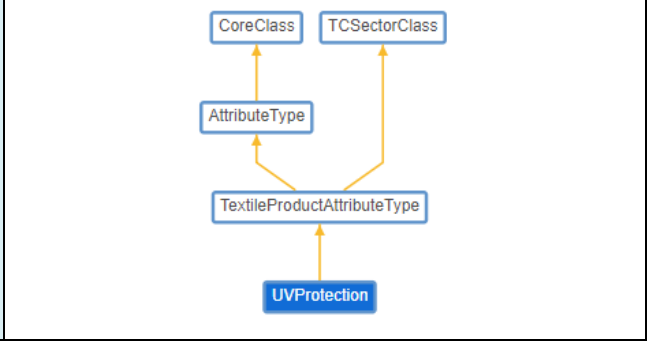
Taxonomy	Graph visualisation
	
	
	

Table 13 Attribute Type taxonomy in Textile-Cosmetics Sector

5.3 Cross-Sector concepts

5.3.1 Certifications

Table 14 shows taxonomy of certifications.

Taxonomy	Graph visualisation
AnySectorClass Certification CompanyCertification LEED QualityCertification AS AS9000 AS9100 ISO ISO14000 ISO9000 ISO9001 ISO9002 ISO9003 ISO9004 MIL QS9000	

Table 14 Taxonomy of certifications

5.3.2 Industries

Table 15 shows an initial taxonomy of classification of industries. This taxonomy is directly reused from a MSDL ontology (Ameri and Dutta 2006). It is not based on any standardised taxonomy of industry classification at this stage, but nonetheless, the standardized classifications of industries, for example NAISC, can be imported into the ecosystem database with minor effort.

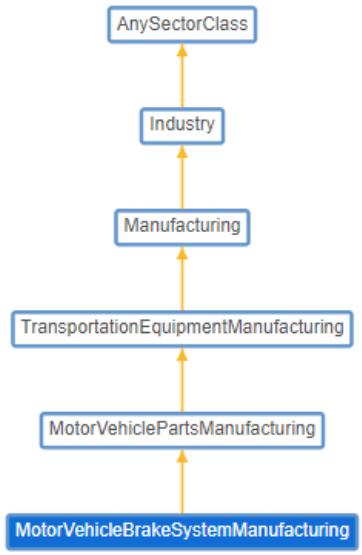
Taxonomy	Graph visualisation
AnySectorClass Certification Industry Manufacturing ChemicalManufacturing ComputerAndElectronicProductManufacturing ElectricalEquipmentAppliance AndComponentManufacturing FabricatedMetalProductManufacturing FoodManufacturing MachineryManufacturing MiscellaneousManufacturing PrimaryMetalManufacturing TransportationEquipmentManufacturing AerospaceProductandPartsManufacturing MotorVehicleBodyandTrailerManufacturing MotorVehicleManufacturing MotorVehiclePartsManufacturing MotorVehicleBrakeSystemManufacturing MotorVehicleElectricalAndElectronicEquipmentManufacturing MotorVehicleGasolineEngineAndEnginePartsManufacturing MotorVehicleMetalStamping MotorVehicleSeatingAndInteriorTrimManufacturing MotorVehicleSteeringAndSuspensionComponentsExceptSpringManufac MotorVehicleTransmissionAndPowerTrainPartsManufacturing OtherMotorVehiclePartsManufacturing OtherTransportationEquipmentManufacturing RailroadRollingStockManufacturing ShipandBoatBuilding Textile	

Table 15 Taxonomy of industrial classification

5.3.3 Human Capabilities

Table 16 shows taxonomy of classification of human capabilities non-related to specific manufacturing sector.

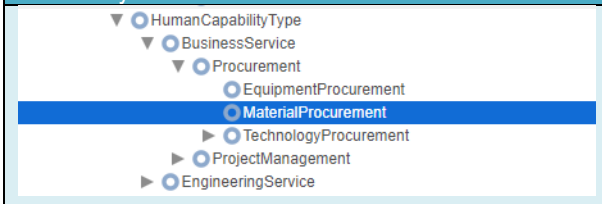
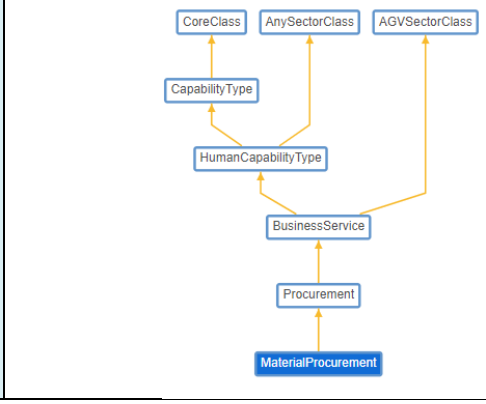
Taxonomy	Graph visualisation
	

Table 16 Classification of human capabilities non-related to specific manufacturing sector

Please note that there are some existing classification systems that can be integrated into MANU-SQUARE in a future. For example, ESCO[7] (European Skills/Competences, Qualifications and Occupations) is a multilingual classification system covering skills, competences, qualifications and occupations. Or, Occupational Information Network (O*NET) is a free online database that contains occupational definitions. For each job, O*NET[8] provides the list of skills and knowledge required to perform the work.

[7] https://ec.europa.eu/esco/portal/home
[8] https://www.onetonline.org/

6 MANU-SQUARE ONTOLOGY DEVELOPMENT TOOL

This section presents the MANU-SQUARE ontology development tool. This tool will allow domain-experts and platform managers to facilitate further development, expansion and evolution of the domain-specific ontologies in MANU-SQUARE. In addition to the ontology editor, it provides inference rules editor. The tool builds on MANU-SQUARE ontology development methodology, with controlled templates for capturing domain-specific concepts and their taxonomies, and RDFS/OWL representation and verification of ontologies. The section starts with the requirements specification, the outlines the tool design, and finally provide the implementation details.

6.1 Requirements

In the systems and requirements engineering, a common recommendation to classify requirements is by categorizing them either as functional or as non-functional. In short, functional requirements describe what the system does in terms of its functions, where a function consists of an input, output, and of a behaviour that translates the input to output, and directly addresses user needs. On other side, non-functional requirements are operational characteristics to judge the system prior, during and after the execution of functions. Therefore, the requirements related to MANU-SQUARE ontology authoring tool are systematized into functional and non-functional ones, and prioritized accordingly. Each requirement has:

- Requirement ID – an unique identifier of requirement
- Requirement Description – Explaining the requirement including when applicable, the source and rationale.
- Requirement Priority – relative importance among all the requirements. To rank and prioritize requirements we distinguish three classes of priorities - essential, conditional, and optional - according to IEEE 830-1998 recommendation.
    - Essential - implies that the software will not be acceptable unless these requirements are provided in an agreed manner.
    - Conditional - Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent.
    - Optional - Implies a class of functions that may or may not be worthwhile.

  In this document, requirements are prioritized using only 'essential' and ‘conditional’ priorities. 'Optional' features are out of our consideration.

Table 17 summarises the functional requirements, while Table 18 summarizes non-functional requirements.

Requirement ID	Requirement description	Priority
FR_01	Editing -- Creation and deletion of classes, properties and instances in/from domain-specific ontologies and application-specific ontologies.	Essential
FR_02	Editing -- Tree-like visualisation of classes hierarchy (i.e. of taxonomies)	Essential
FR_03	Editing -- Basic refactoring ability – refactoring to resource names	Essential
FR_04	Editing --Tool-embedded enforcement of MANU-SQUARE ontology design recommendations (ODPs) and adherence/enforcement to the MODE (MANU-SQUARE domain-specific ontology methodology). • Graphical user interface templates for capturing extensions of different taxonomies include ProcessType, CapabilityType, EquipmentType, ProductType, ByProductType, WasteTypes, MaterialTypes, AttributeTypes, Certification, Industry classification taxonomies. • Graphical user interface template for introduction of new SectorClasses, which serve to categories different concepts and taxonomies into their respective industrial sectors. • Strict enforcement of ontology design principles: o each new domain specific concept must be a sub-	Essential

	sumption of a MANU-SQUARE core concept ○ each new domain specific concept must categorized into specific-SectorClass, either sector-specific or cross-sector one. ○ each new domain specific concept must have rdfs:label annotation. This annotation can be used for text-based search. ○ each new domain specific concept is disjoined from existing concepts ○ each new domain-specific property must be a subproperty of MANU-SQUARE core property	
FR_05	Editing --- autocomplete support e.g. autocompletion of class names in a subclass statement definitions.	Conditional
FR_06	Semi-automated support – tool should provide means for recognition of entities from textual-descriptions related to MANU-SQUARE domains. The recognized entities should not be automatically added into domain-specific ontologies, as they must be reviewed by the experts, thus, rather should be approved and introduced by the domain experts.	Conditional
FR_07	Semi-automated support – tool should be able to lookup for an external ontological resources in order to find and reuse existing formalized domain knowledge. For example, it may query KbPedia database (http://kbpedia.org/) to retrieve external knowledge / formalisation of MANU-SQUARE concepts.	Conditional
FR_08	Import ontologies – tool must be able to import domain-specific OWL/RDFS ontologies from files.	Essential
FR_09	Import rules – tool must be able to import inference rules from text files.	Essential
FR_010	Export Ontologies – the tool must be able to export ontologies into different RDF/OWL formats, including RDF/XML, N3, Turtle, etc.	Essential
FR_011	Export Ontologies – the tool must be able to export inference rules into appropriate formats.	Essential
FR_012	Integration with semantic database – ontologies must be saved into sematic infrastructure of the platform (RDF/RDFS/OWL databases)	Essential
FR_013	Test rules – tools must provide to the user ability to test the inference rules i.e. to verify their syntax	Essential
FR_014	Consistency check – tool must integrate open-source, free-licence OWL reasoners for check of consistency of domain ontologies	Essential
FR_015	Querying tab -- tool should provide editor to execute SPARQL queries on domain-specific ontologies and semantic infrastructure.	Conditional
FR_016	Visualisation – tool should provide user-friendly visualisation of domain taxonomies graphs (in a manner similar to Web Protégé tool)	Conditional
FR_017	Visualisation – ontological resources can be visualised/labelled using either their IRI or language-tagged rdf:labels	Essential

Table 17 Ontology authoring tool – Functional requirements

Requirement ID	Requirement description	Priority
NFR_01	Technology -- tool has to be developed as a web application, as also other front-end app of the platform will be based on web technologies and web GUIs.	Essential
NFR_02	UX -- tool should be intuitive and user-friendly.	Essential
NFR_03	UX -- tool should provide meaningful messages to the users (e.g., in a case of user errors, application errors, server failures, etc.)	Essential
NFR_04	Security -- tool has to be secured. On other words, users must authenticate themselves before accessing to the tool features. The domain ontologies and rules are important for the platform functionalities, thus, any unauthorized change of the ontologies and the	Essential

	inference rules may break down functionalities. Authorization must also be in place, with at least two set of permission -- read/write of ontologies and rules.	
NFR_05	Performability --- tool must be able to provide appropriate response and processing times and throughput rates when performing functions.	Essential
NFR_06	Scalability – scalability aspect should also be taken into consideration. Scalability is a capability of a software system to handle a growing number of users and data with minimal impact to costs and resources.	Essential
NFR_07	Reusability – to reduce the development and deployment costs, tools should be using appropriate (of good quality) existing services and data sources, and should be using already proven and tested technologies.	Essential
NFR_08	Openness -- open source technologies: technologies to implement the tool should be open source and licence-free, to adhere to the rest of platform.	Essential

Table 18 Ontology authoring tool – Non-functional requirements

6.2 Design (Internal architecture)

The tool design and internal architecture supports the requirements introduced in § 6.1. Internal architecture is outlined in Figure 21.

Figure 21 MANU-SQUARE ontology authoring tool – internal architecture

The main subcomponents (services) of the tool are:

- **Front-end** component. Front-end component is a web-based graphical user interface (GUI) for usage of various features of the service layer.
- **Service** layer
  - **Ontology Importer Manager**. A component is responsible for handling an import of ontologies from the file system. Imported ontologies must adhere to the proposed design and architectural structure of MANU-SQUARE ontologies. Once imported, they can be then further edited, integrated with existing ontologies in the ecosystem, and persisted into a database.
  - **Ontology Editor Manager**. A component that provides dialogs for creation of ontology classes, properties and instance. It manages templates for capturing extensions of domain-specific taxonomies i.e. for ProcessType, CapabilityType, EquipmentType, ProductType, ByProductType, WasteTypes, MaterialTypes, AttributeTypes, Certification, Industry classification taxonomies. Then, it provides a simple template for introduction of SectorClasses, which serve to categorise concepts and taxonomies into their respective industrial sectors. Further, Editor Manager enables simple refactoring of resources

(e.g. renaming of resources) and text autocomplete features (e.g. autocompletion of class names for subclass statements definition). The editor manager interacts with the semantic database to provide for retrieval, update, and delete operations on existing ontologies, and for insert new ontologies operations.

- **Rules Editor Manager**. A component that responsible to store and retrieve inference rules, in their native form and syntax, to/from the semantic database.
- **Querying Manager**. A component responsible for execution of SPARQL queries on ontological resources. It accepts SPARQL query in its native form, sends it to a query processor, and accepts the result. The query processor is an external component (it might be provided by the semantic infrastructure or elsewhere). Several examples of SPARQL queries are in Appendix C.
- **Ontology Export Manager**. A component responsible for export of ontologies into files. Export can be done in several formats, including RDF/XML, N3, Turtle, etc.
- **Reasoning / Consistency check Manager**. This manager component integrates a reasoning engine that provides a consistency check of ontology. Free and open source reasoners are available (e.g. JFact, HermiT, Pellet)
- **Entity recognition /extraction Manager.** This components provides analysis the textual descriptions of manufacturing resources in order to recognize named entities from the text and to classify them into pre-defined categories (i.e. into MANU-SQUARE core concepts and their extensions).
- **Knowledge Integration and Reusability Manager.** This component is responsible to search for MANU-SQUARE-relevant knowledge over public knowledge bases. The aim is to reuse already formalized knowledge and to integrate it with MANU-SQUARE ontologies, either by importing the concepts from public knowledge base into the ecosystem or by linking those external concepts to MANU-SQUARE's ones. In either case, the process should be controlled by the domain expert i.e. the domain expert should verify and approve concepts before they get imported in or linked with MANU-SQUARE ontologies. For examples, KBpedia, is a public knowledge base, written primarily in OWL 2, includes 55,000 reference concepts, about 30 million entities, and 5,000 relations and properties, all organized according to about 70 modular typologies that can be readily substituted or expanded (http://kbpedia.org/).
- **Ontology Visualisation Manager**, responsible for a graph-based visualisation of taxonomies.

- **Semantic database** is a transactional RDF triplestore built on RDF, RDFS and OWL standards.

6.3 Implementation

The tool is implemented using Java technology stack. Apache Jena API (https://jena.apache.org/) is used to for handling RDF, RDFS, OWL data and models. Spring (https://spring.io/) is used for implementation of back-end service layer that is based on model-view-controller design for web applications. Spring also provides needed security features (authentication and authorisation) for a service-layer. Primefaces (https://www.primefaces.org/) is used for development of front-end components and controls. The tools is a web Java-based application and requires Apache Tomcat container. In the following figures, illustrative screenshots of the tool are shown. The source code is hosted on a GitHub version control system -- https://github.com/vujasm/mode

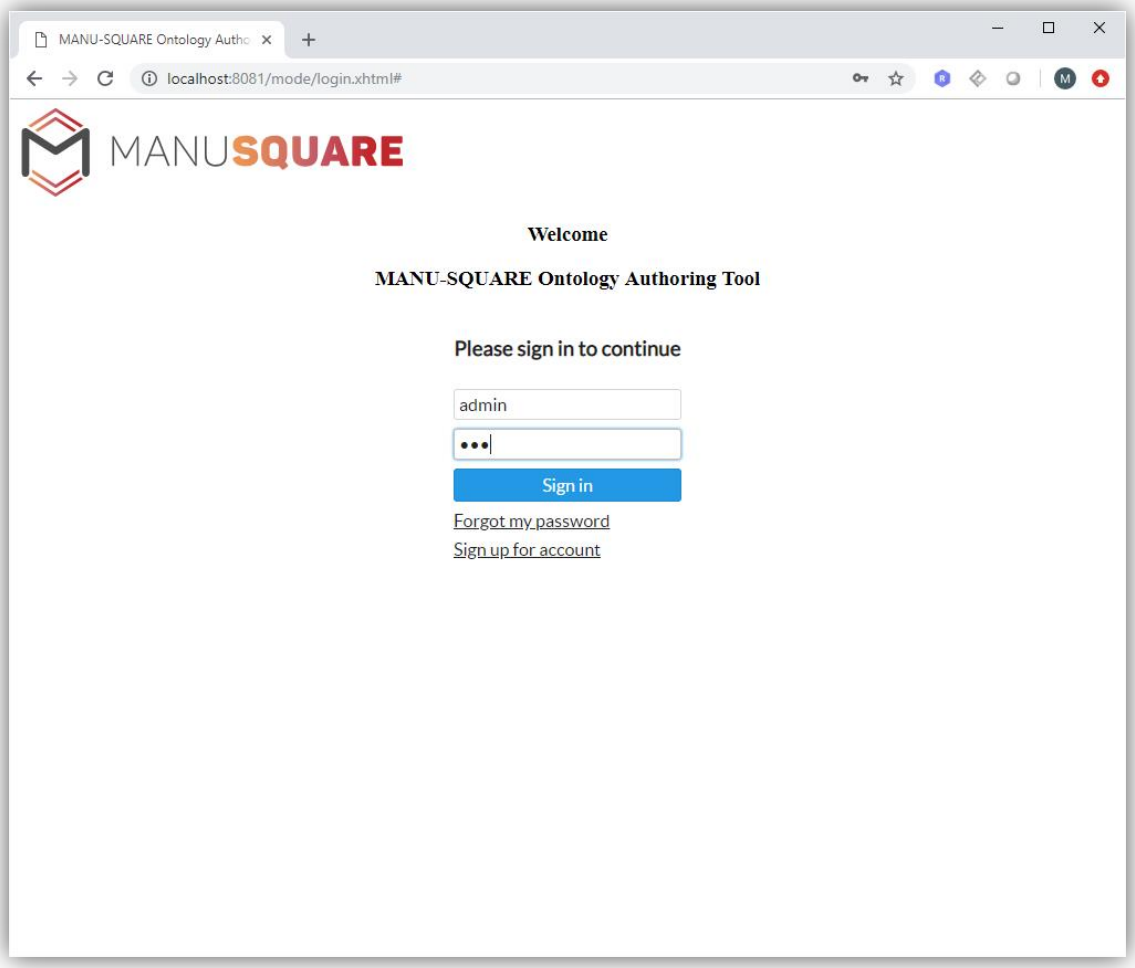


Figure 22 MODE Ontology Authoring Tool – Login screen

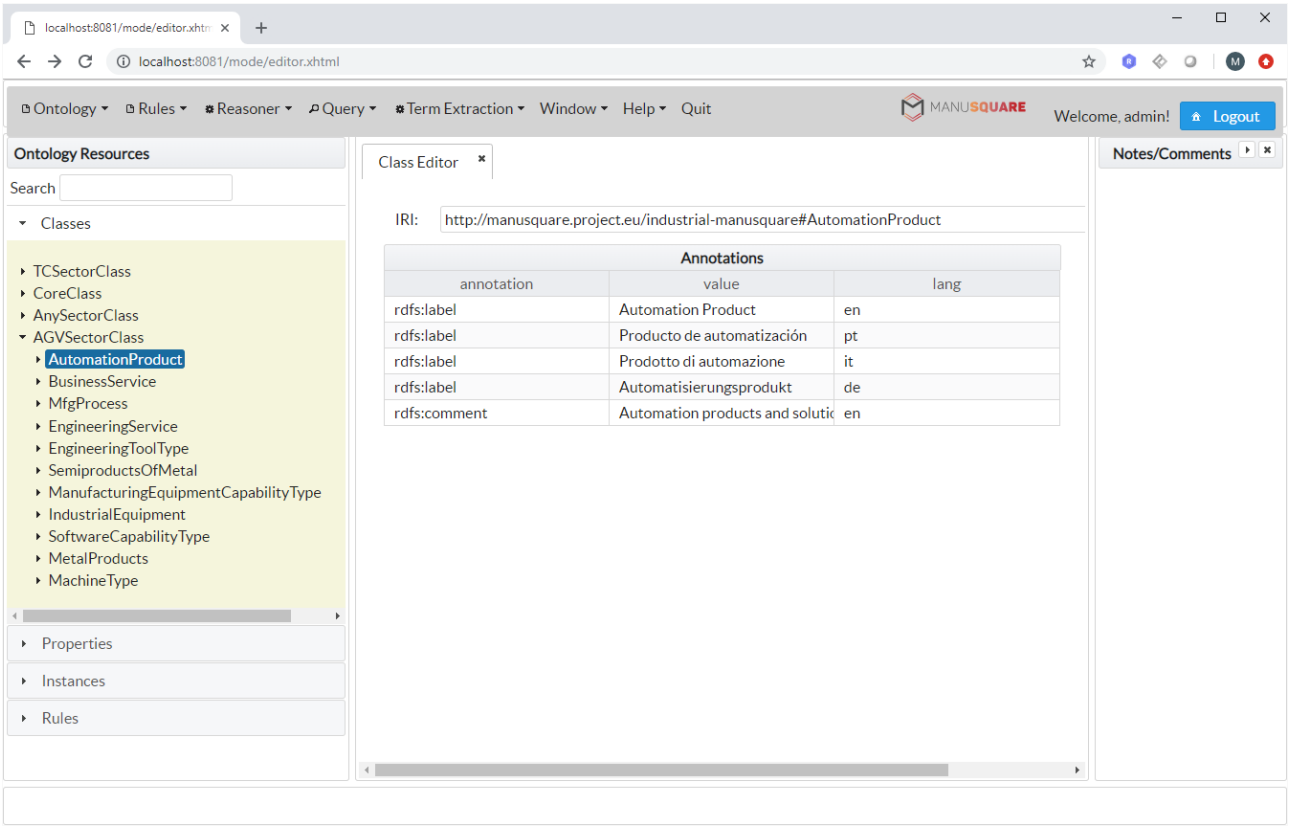


Figure 23 MODE Ontology Authoring Tool – Home screen/Class Editor

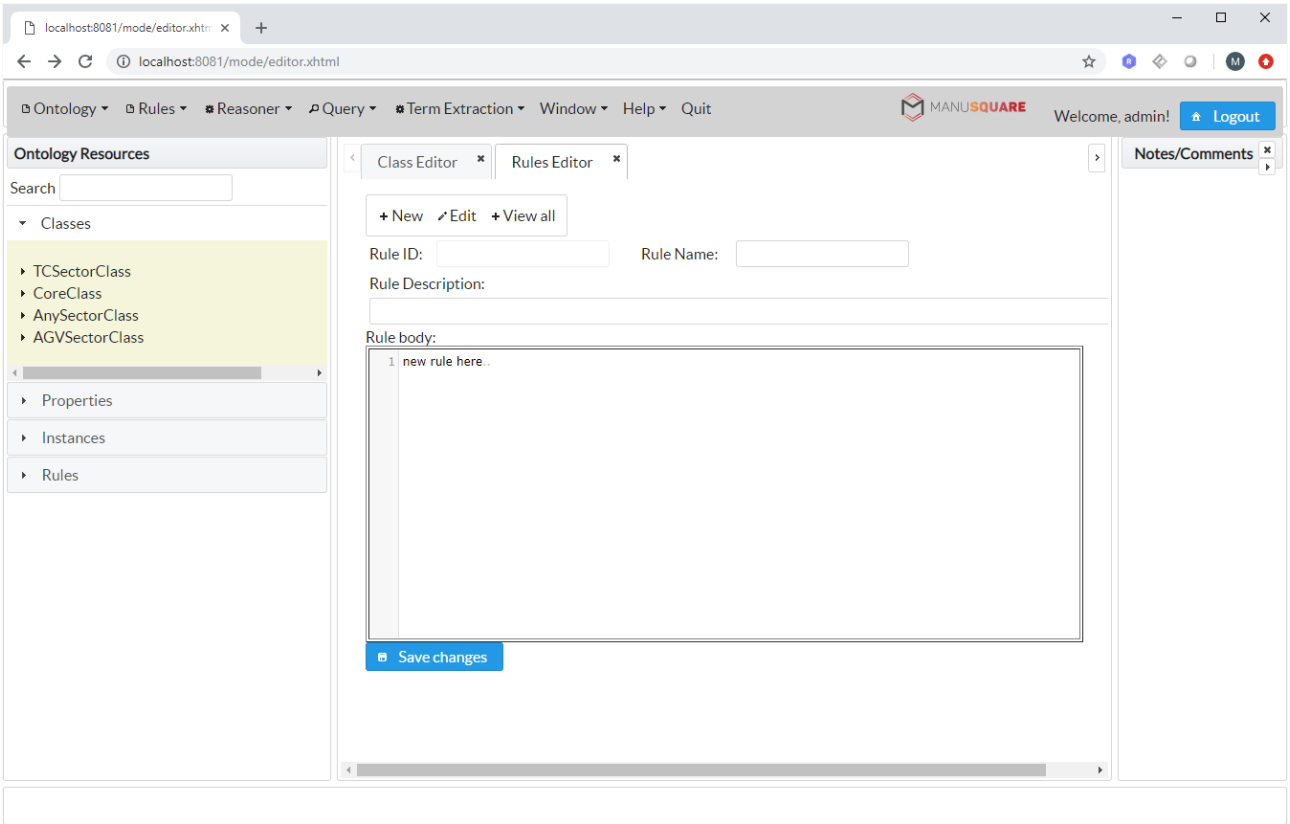


Figure 24 MODE Ontology Authoring Tool – Rules Editor screen

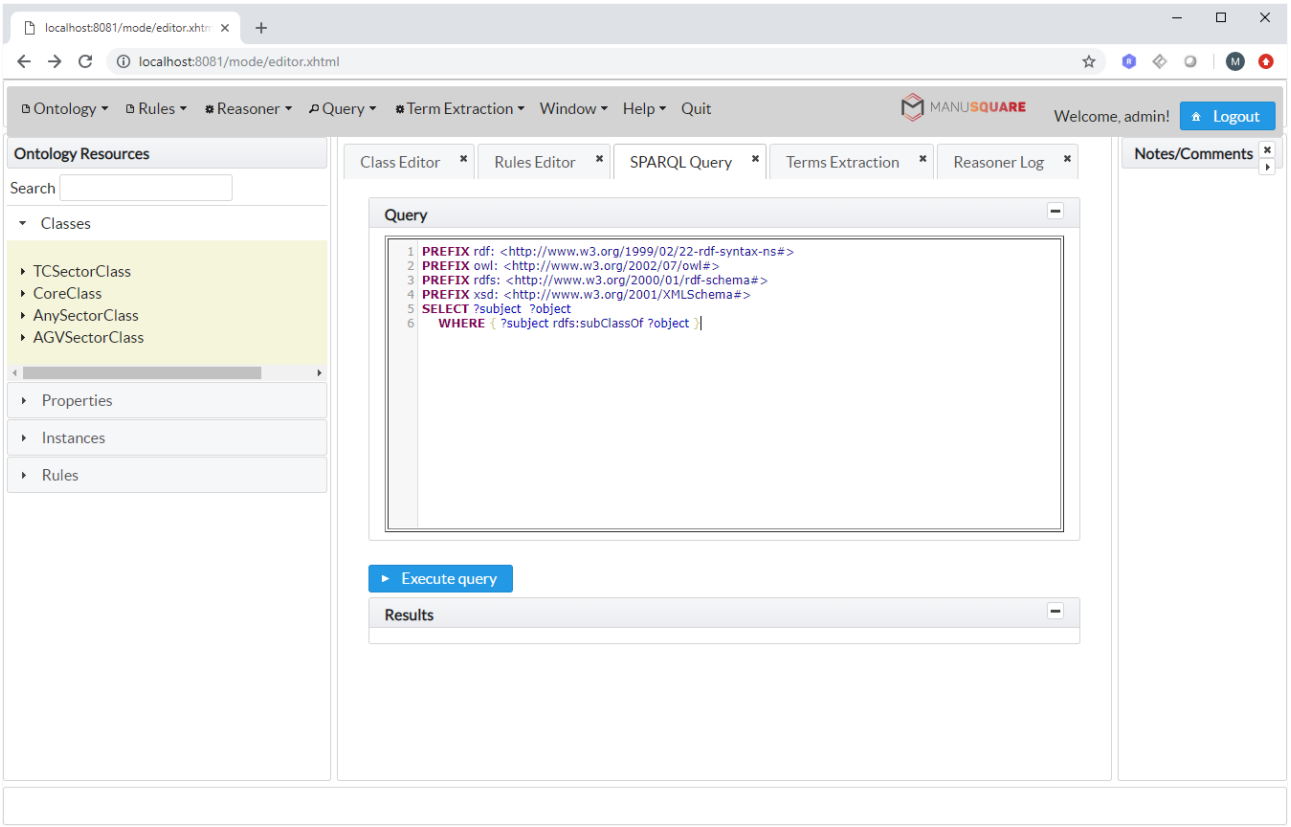


Figure 25 MODE Ontology Authoring Tool – SPARQL Query screen

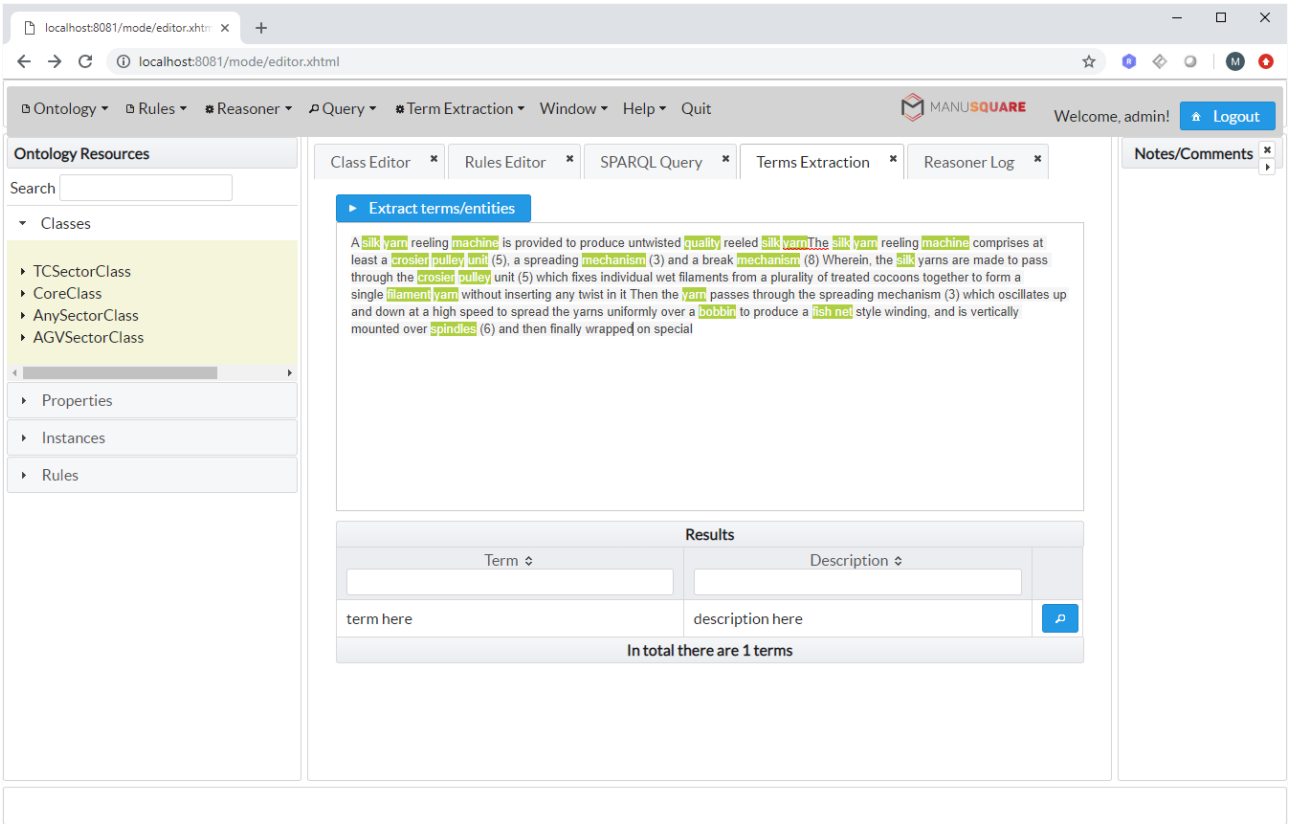


Figure 26 MODE Ontology Authoring Tool – Terms Extraction screen

7 CONCLUSION AND NEXT STEPS

The main goal of this deliverable was to introduce the domain-specific ontologies as domain-specific extension of MANU-SQUARE core concepts to facilitate execution of two MANU-SQUARE demonstration scenarios (scenarios are outlined in D1.3). Another relevant goal was to introduce an ontology authoring tool that shall help to the platform managers and domain experts to: (a) facilitate development and evolution of the domain-specific extension of core concepts when platform further expands within existing or enters new sectors, and (b) to provision area-specific models and inference rules.

The domain-specific ontologies have been developed and explained, along with a first set of ontology design rules that provide guidelines and recommendations for the MANU-SQUARE domain-specific ontologies development. The adherence of MANU-SQUARE domain-specific ontologies to ontology design rules is important for the many reasons, as discussed thoroughly in this deliverable. If the design does not follow a strict design principles/recommendations, the ontologies may become very complex, hard to read, refactor, maintain, debug, evolve, and unusable for the matchmaking operators.

Further, the deliverable presented an early version of the MANU-SQUARE's ontology editor tool that supports development of taxonomies of the core concepts such as process types, equipment types, capability types, product types, by-product types etc. The tool has standard features including imports of ontologies, possibility to export ontologies in different formats, integration with database repository, and integration with semantic reasoners that perform consistency checks of ontologies.

The outcomes described in this deliverable shall be further elaborated in Task 2.3 and Task 2.4. Task 2.3 focuses on an application-specific models and inference rules. The ontology editor in a current version does not provide provisioning of application-specific models and inference rules, thus, these additional features shall be provided in a next release of the tool, which is planned for M16. Application-specific models, their scope and richness of the model expressivity, as well as needed inference rules, have not been described yet in the details that would be sufficient to design and implement the tool-support. Hence, the reason for not providing a tool-support for provisioning of application-specific models and inference rules at this stage of the project is very practical. Nonetheless, these features shall be provided in a next release.

As previously mentioned, the tool uses a third-party Entity Extraction API that is able to detect concepts from the short text using knowledge structured on DBpedia and Wikipedia. Yet, a further enhancement and validation of support for semi-automated knowledge elicitation is left for the next iteration, as before any automated process takes place, the developed domain-specific ontologies should be tested and evaluated by the early prototypes of the tools (notably, by the matchmaking tool). Task 2.4 will deploy a semantic infrastructure and the MANU-SQUARE's ontology editor will be integrated with it, hence, further enhancements of the editor will be provided within the scope of Task 2.4.

REFERENCE LIST

Abburu, Sunitha, and G. Suresh Babu. "Survey on ontology construction tools." International Journal of Scientific & Engineering Research, IV (2013): 1748-1752.

Alatrish, E. S. "Comparison some of ontology." Journal of Management Information Systems 8.2 (2013): 018-024.

Ameri, Farhad, and Debasish Dutta. "An upper ontology for manufacturing service description." ASME 2006 international design engineering technical conferences and computers and information in engineering conference. American Society of Mechanical Engineers, 2006.

Aranguren, M. E., Antezana, E., Kuiper, M., & Stevens, R. (2008). Ontology Design Patterns for bio-ontologies: a case study on the Cell Cycle Ontology. BMC Bioinformatics, 9.

Corcho, Oscar, et al. "Building legal ontologies with METHONTOLOGY and WebODE." Law and the semantic web. Springer, Berlin, Heidelberg, 2005. 142-157.

De Nicola, Antonio, Michele Missikoff, and Roberto Navigli. "A proposal for a unified process for ontology building: UPON." International Conference on Database and Expert Systems Applications. Springer, Berlin, Heidelberg, 2005.

Fernández-López, Mariano. "Overview of methodologies for building ontologies." (1999).

Gamma, E., Helm, R., Ralph, J., & Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software. Addison Wesley.

Gangemi, A. (2005). Ontology design patterns for semantic web content. Proceedings of the Fourth International Semantic Web Conference (pp. 262--276). Berlin: Springer.

Gangemi, A., & Chaudhri, V. (2009). Representing the Component Library into Ontology Design Patterns. Workshop on Ontology Patterns International Semantic Web Conference.

Grüninger, Michael, and Mark S. Fox. "The role of competency questions in enterprise engineering." Benchmarking—Theory and practice. Springer, Boston, MA, 1995. 22-31.

Iqbal, Rizwan, Masrah Azrifah Azmi Murad, Aida Mustapha, and Nurfadhlina Mohd Sharef. "An analysis of ontology engineering methodologies: A literature review." Research journal of applied sciences, engineering and technology 6, no. 16 (2013): 2993-3000.

Fernández-López, Mariano, Asunción Gómez-Pérez, and Natalia Juristo. "Methontology: from ontological art towards ontological engineering." (1997).

Grüninger, Michael, and Mark S. Fox. "Methodology for the design and evaluation of ontologies." (1995).

Uschold, Michael, and Martin King. "Towards a methodology for building ontologies." (1995).

Uschold, Mike, and Michael Gruninger. "Ontologies: Principles, methods and applications." The knowledge engineering review 11.2 (1996): 93-136.

Manchester ODP Library. (2009, Jul). Retrieved March 25, 2013, from Manchester Ontology Design Patterns: http://www.gong.manchester.ac.uk/odp/html/index.html

Neon ODP Library. (2012, May). Retrieved March 25, 2013, from Neon Ontology Design Patterns: http://ontologydesignpatterns.org/

Noy, Natalya F., and Deborah L. McGuinness. "Ontology development 101: A guide to creating your first ontology." (2001).

Pieterse, Vreda, and Derrick G. Kourie. "Lists, Taxonomies, Lattices, Thesauri and Ontologies: Paving a Pathway Through a Terminological Jungle." Knowledge Organization 41.3 (2014).

Presutti, V., Gangemi, A., David, S., de Cea, G. A., Surez-Figueroa, M., Montiel-Ponsoda, E., et al. (2008). D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NEON project deliverable.

Rector, A. (2005, May). Representing Specified Values in OWL: "value partitions" and "value sets". Retrieved March 2013, from W3C - Semantic Web Best Practices and Deployment Working Group: http://www.w3.org/TR/swbp-specified-values/

Slimani, Thabet. "Ontology development: A comparing study on tools, languages and formalisms." Indian Journal of Science and Technology 8.24 (2015).

Suárez-Figueroa M.C., Gómez-Pérez A., Fernández-López M. (2012) The NeOn Methodology for Ontology Engineering. In: Suárez-Figueroa M., Gómez-Pérez A., Motta E., Gangemi A. (eds) Ontology Engineering in a Networked World. Springer, Berlin, Heidelberg

Suarez-Figueroa, M., Gomez-Perez, A., Motta, E., & Gangemi, A. (2012). Introduction: Ontology Engineering in a Networked World. London: Springer.

Sure, York, Steffen Staab, and Rudi Studer. "On-to-knowledge methodology (OTKM)." Handbook on ontologies. Springer, Berlin, Heidelberg, 2004. 117-132.

Swartout, B., R. Patil, K. Knight and T. Russ, 1996. Toward distributed use of large-scale ontologies. Proceeding of the 20th Workshop on Knowledge Acquisition for Knowledge-Based Systems, pp: 138-148.

T. R. Gruber (1993). A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5, pp. 199-220.

Usman, Zahid, et al. "A manufacturing core concepts ontology for product lifecycle interoperability." International IFIP Working Conference on Enterprise Interoperability. Springer, Berlin, Heidelberg, 2011.

Vujasinovic, Marko, Nenad Ivezic, and Boonserm Kulvatunyou. "A survey and classification of principles for domain-specific ontology design patterns development." Applied Ontology 10.1 (2015): 41-69.

W3C. (2005, March). Semantic Web Best Practices and Deployment Working Group. Retrieved March 25, 2013, from W3C: http://www.w3.org/2001/sw/BestPractices/

Youn, Seongwook, and Dennis McLeod. "Ontology development tools for ontology-based knowledge management." Encyclopedia of E-Commerce, E-Government, and Mobile Commerce. IGI Global, 2006. 858-864.

APPENDIX A - KNOWLEDGE ELICITATION TEMPLATE FOR DOMAIN EXPERTS (JPM'S FEEDBACK)

Please complete the following table by listing the key concepts (e.g. productA, productB, etc.) and/or providing brief descriptions (e.g. shortly describing a use-case relevant process, its input/output, involved resources, etc.) that specialise the indicated reference elements (coming from the core model of MANU-SQUARE introduced in the previous sections) into your specific use case.

USE-CASE	New automated guided vehicle (AGV) for the food processing industry
Indicate the **production / machining processes** in your factory	For this demonstration scenario, the applicable processes are engineering processes: - Concept definition and validation; - Layout of equipment and implementation area definition; - FMEA analysis; - Acceptance criteria and validation criteria definition; - Product development and design – 3D design; - BoM definition and construction; - Industrialization of the project – 2D design + data sheets + flat patterns - Procurement for new applicable materials and technologies;
Indicate the **products** that you make	Industrial conveyors (individual packages, group packages, pallets, bulk materials, etc.), automation equipment (control cabinets, electrical cabinets, AGV's, etc.), industrial equipment (various applications, and functionalities), robotics (palletizing, depalletizing, pick and place), software (HMI, Supervision systems, SCADA systems, etc.), turn-key industrial projects (from project design, to equipment choosing and implementation, design and implementation of utilities, etc.), retrofitting services, preventive and curative maintenance.
Indicate the **material/sub-products** that are needed for your production	On the demonstration scenario perspective, they would be engineering activities. - Equipment development – structure, functionalities, design, materials, etc. - 2 D drawing planification; - BoM construction and definition.
Indicate **by-products produced** in your factory	Usually no by-products that are able to be directly reused.
Indicate **machining equipment/tools** that your production has / needs. Indicate **capabilities** of the tools	Engineering activities that can be subcontracted: Product development and design – 3D design; BoM definition and construction; Industrialization of the project – 2D design + data sheets + flat patterns
Indicate **KPIs** used to assess the processes, products and resources listed above	Quality of the processes and products, lead time, price, skills and competences.
Certifications of your company	ISO 9001:2015, ISO 14001:2015, OSHAS 18001:2007, NP 4427 (Portuguese standard for Human Resources management).
Sectors served	Food & Beverage, Chemical, Pharmaceutical, Logistics (post and parcel, etc.), Industrial companies with intralogistics processes.
Provide **BOM** for **automated guided vehicle (AGV)**	TBD – BoM is an outcome of the development process.
Provide **operations and routings list** for **automated guided vehicle (AGV)** production	TBD – Operations and routings list are an outcome of the development process.

USE-CASE	Retrofitting services
Indicate the **production / machining processes** in your factory	Retrofitting services, that can be contracted by customers on the platform consist on: - Equipment evaluation and technical assessment; - Requirements list : materials, equipment, tasks for successful retrofitting; - Retrofitting planning; - Retrofitting services – onsite (customer) or at JPM facilities;

	- *Preventive and Curative equipment maintenance – complement for retrofitting services.*
*Indicate the **products** that you make*	*Industrial conveyors (individual packages, group packages, pallets, bulk materials, etc.), automation equipment (control cabinets, electrical cabinets, AGV's, etc.), industrial equipment (various applications, and functionalities), robotics (palletizing, depalletizing, pick and place), software (HMI, Supervision systems, SCADA systems, etc.), turn-key industrial projects (from project design, to equipment choosing and implementation, design and implementation of utilities, etc.), retrofitting services, preventive and curative maintenance.*
*Indicate the **material/sub-products** that are needed for your production*	*Not applicable on this scenario (JPM will be the supplier).*
*Indicate **by-products produced** in your factory*	*Usually no by-products that are able to be directly reused.*
*Indicate **machining equipment/tools** that your production has / needs.* *Indicate **capabilities** of the tools*	*Not applicable on this scenario (JPM will be the supplier).*
*Indicate **KPIs** used to assess the processes, products and resources listed above*	*Quality of the processes and products, lead time, price, skills and competences.*
Certifications of your company	*ISO 9001:2015, ISO 14001:2015, OSHAS 18001:2007, NP 4427 (Portuguese standard for Human Resources management).*
Sectors served	*Food & Beverage, Chemical, Pharmaceutical, Logistics (post and parcel, etc.), Industrial companies with intralogistics processes.*
*Provide **BOM** for **automated guided vehicle (AGV)***	*Not applicable for this scenario.*
*Provide **operations and routings list** for **automated guided vehicle (AGV)** production*	*Not applicable for this scenario.*

Please send us any additional material (pdf, excel, online references, etc) that could help to identify different types of machining processes, materials/parts, machine tools and equipment that is used in your respective domain (machining).
Moreover, if you are aware or if you already make use of some taxonomy/classification/standards for some of the aspects listed above (e.g. for describing/tagging your products/processes in some commercial/business directories) please send us any reference.

APPENDIX B – KNOWLEDGE ELICITATION TEMPLATE FOR TOOL DEVELOPERS (SUSPI'S FEEDBACK)

Tool Name	SUSTAINABILITY ASSESMENT TOOL and UNIFIED FLOW MANAGER
What kind of queries (i.e. Competency Questions) the semantic infrastructure should be able to answer?	*-) retrieve all process by company (e.g. Company S.A.)* *-) retrieve all process by input (e.g. steel)* *-) retrieve all process by output (e.g. steel)* *-) retrieve all process by resources (e.g. milling equipment…)* *-) retrieve all resources by attributes (e.g. power 100 W)* *-) retrieve all companies by process (e.g milling)* *FILTERS like > or < or = are applied on each query* *SORTING STRAGIES*: order by asc, desc* *-) retrieve all equipment with the same attributes (eg. attribute power)* *-) retrieve all item using the same component* *-) retrieve all processes using the same input* *-) retrieve all processes using the same output* *-) retrieve all the companies using the same processe*
Do you need to extend the core model (D2.1) with additional concepts that are not currently included? If so, provide more details.	*NO*

Explanation note: We are looking to get an initial list of example questions that a knowledge (semantic) base based on the MANU-SQUARE ontologies should be able *to answer. We refer to those questions as competency questions. The competency questions (CQs in short) are example questions that MANU-SQUARE ontologies should be able to answer. CQs are usually provided in a natural language form and are very useful to determine the scope and competency of the ontology.*

For example, competency questions, generic though, might be:

- *Who are suppliers with some specific production capacity e.g. production of steel sheets?*
- *Who are suppliers with some specific capability e.g. cnc machining or laser cutting?*
- *Who are suppliers with some specific equipment e.g. CNC milling?*

Or, examples of CQs for answering about suppliers offering specific capabilities (CNC machining), with specific dimensional properties (length, diameter):

- *Which suppliers provide a CNC laser machining?*
- *Which suppliers provide a CNC laser machining with 4-axis?*
- *Which suppliers provide a CNC laser micro machining?*
- *Which suppliers provide a CNC machining of part with certain length and diameter size?*
- *Which suppliers provide a CNC machining of part with certain length and diameter size AND can do stress reliving as well as prototyping?*
- *Which suppliers are specialized in CNC laser micro machining for metals?*

Further, CQ could be

- *Which companies produce certian by-waste (e.g. silk) ?*
- *How much energy (kwatt) uses CNC Milling Machine of supplier XYZ?*

APPENDIX C – SPARQL EXAMPLES

This appendix shows several examples of SPARQL queries that are based on MANU-SQUARE Core ontology and its domain-specific extensions.

Example 1. Retrieve all processes by supplier name e.g. "JPM"

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://MANU-SQUARE.project.eu/core-MANU-SQUARE#>
PREFIX ind: <http://MANU-SQUARE.project.eu/industrial-MANU-SQUARE#>
SELECT     ?process ?supplierName
            WHERE { ?processChain core:hasSupplier ?supplier .  ?supplier core:hasName ?supplierName.
            FILTER regex(?supplierName, "JPM")
            ?processChain core:hasProcess ?process. ?process rdf:type ?processType }
```

Example 2. Retrieve processes and process types by manufacturer's name e.g. "JPM"

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://MANU-SQUARE.project.eu/core-MANU-SQUARE#>
PREFIX ind: <http://MANU-SQUARE.project.eu/industrial-MANU-SQUARE#>
SELECT     ?process ?processType ?supplierName
            WHERE { ?processChain core:hasSupplier ?supplier .  ?supplier core:hasName ?supplierName.
            FILTER regex(?supplierName, "JPM")
            ?processChain core:hasProcess ?process. ?process rdf:type ?processType }
```

Example 3. Retrieve processes by process type (e.g. CNCMilling)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://MANU-SQUARE.project.eu/core-MANU-SQUARE#>
PREFIX ind: <http://MANU-SQUARE.project.eu/industrial-MANU-SQUARE#>
SELECT distinct ?supplierName ?process ?processType
            WHERE { ?processChain core:hasSupplier ?supplier .  ?supplier core:hasName ?supplierName.
            ?processChain core:hasProcess ?process. ?process rdf:type ?processType.
            FILTER(?processType IN (ind:CNCMilling))}
```

Example 4. Retrieve processes by input (e.g. Steel)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://MANU-SQUARE.project.eu/core-MANU-SQUARE#>
PREFIX ind: <http://MANU-SQUARE.project.eu/industrial-MANU-SQUARE#>
SELECT distinct ?supplierName ?process ?processType ?materialType
            WHERE { ?processChain core:hasSupplier ?supplier .  ?supplier core:hasName ?supplierName.
            ?processChain core:hasProcess ?process. ?process rdf:type ?processType.
            ?process core:hasInput ?material. ?material rdf:type ?materialType
            FILTER(?materialType IN (ind:Steel))}
```

Example5. Retrieve equipment by capability (e.g. CAD capability)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://MANU-SQUARE.project.eu/core-MANU-SQUARE#>
PREFIX ind: <http://MANU-SQUARE.project.eu/industrial-MANU-SQUARE#>
SELECT ?equipment ?capabilityType
            WHERE { ?equipment core:hasCapability ?capability.
            ?capability rdf:type ?capabilityType.
            FILTER(?capabilityType IN (ind:CAD))}
```