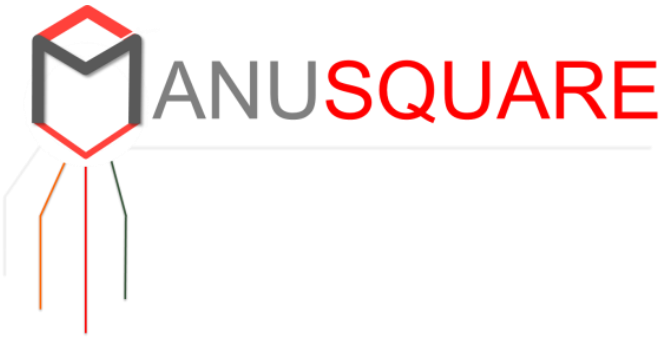
Horizon 2020 – The EU Framework Programme for Research and Innovation
Project Co-funded by the European Commission
Contract number: 761145
Call identifier: NMBP-22-2017
Project Start Date: 1st January 2018



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

D2.1
Meta-models

Dissemination Level	Public
Partners	SUPSI, INNOVA
Authors	Gabriele Izzo, Giuseppe Landolfi, Andrea Barni, Elias Montini, Marko Vujasinovic
Planned date of delivery	M8 – August 2018
Date of issue	29th October 2018
Document version	Final 1.2

DOCUMENT HISTORY

Version	Issue date	Content and changes	Author
0.1	29.06.2018	Table of Content	Giuseppe Landolfi
0.2	13.07.2018	Data model description	Gabriele Izzo, Giuseppe Landolfi, Andrea Barni
0.3	20.07.2018	INNOVA internal review	Alessio Gugliotta, Marko Vujasinovic
0.4	23.07.2018	SUPSI internal review	Gabriele Izzo, Giuseppe Landolfi, Andrea Barni
0.6	16.08.2018	§ 5 – Ontology Implementation description	Alessio Gugliotta, Marko Vujasinovic
0.7	26.08.2018	ToC review	Gabriele Izzo, Giuseppe Landolfi, Andrea Barni
0.8	28.08.2018	§ 4 – Ontology Implementation description review	Alessio Gugliotta, Marko Vujasinovic
0.9	29.08.2018	Executive summary, introduction and conclusion	Gabriele Izzo, Giuseppe Landolfi
1.0	30.08.2018	§ 2.2 – MANU-SQUARE functionalities and ecosystem description	Andrea Barni, Elias Montini
1.1	31.08.2018	Reference list review.	Andrea Barni
1.2	29.10.2018	Quality assurance	Donatella Corti

Role	Partner	Person
Reviewer 1	INNOVA	Alessio Gugliotta
Reviewer 2	INSESCTEC	Antonio Lucas Soares
Quality assurance	SUPSI	Donatella Corti

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Acronym	Description
API	Application Programming Interface
ER	Entity Relationship
ERM	Entity Relationship Modelling
HTTP	HyperText Transfer Protocol
IDEF	Integration DEFinition
KPI	Key Performance Indicator
LCA	Life Cycle Assessment
ORM	Object Role Modelling
RFQ	Request for Quotation
UML	Unified Model Language
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

1 EXECUTIVE SUMMARY

This deliverable documents the first released version of the developed semantic structure of the MANU-SQUARE platform where the core abstract concepts and, consequently, the main propositions of the semantic representation that constitute the backbone structure of the ecosystem data model, have been conceived.

Beside a simple introduction (§ 2), the deliverable follows with a short description of the overall MANU-SQUARE system architecture with the aim to describe the main involved components and their interactions. This chapter ends with a summary of the service platform requirements identified during the activities belonging to Task 1.2.

The § 3 is dedicated to present the existing data modelling techniques in terms of characteristics and applicability in the manufacturing sector. The aim of this section is to give an overview of the standard data model representation alternatives in order to better understand the potentialities in using ontologies.

§ 4 represents the core of the deliverable and starts with a detailed section dedicated to the Ecosystem Data Model formalized to cover all the core abstract concepts that constitute the backbone of the MANU-SQUARE platform. The document then moves to the detailed description of the implemented core ontology (§ 5) starting from the ontology languages overview, going through a list of the ontology representation language choices in other initiatives already existing in manufacturing context and concluding, at the end, to the explanation of the MANU-SQUARE ontology implementation.

Finally, § 6 draws some concluding remarks.

2 INTRODUCTION

This document supports the delivery of the first version of the semantic meta-model for the ecosystem representation within the MANU-SQUARE platform. A detailed definition of the data model has been partially conducted on the base of the results of Task 1.2 "Specification Definition" meant to define in a unique vision specifications, integrating perspectives pertaining to industrial sectors of different nature. The data-modelling phase has been strongly affected by the definition of the value-added services provided by tools belonging to the MANU-SQUARE platform, developed within WP4.

This document has been thought to focus on an overall description of the semantic data model both for the platform tools developers and for the platform stakeholders, thus it has not to be considered as a pure guide.

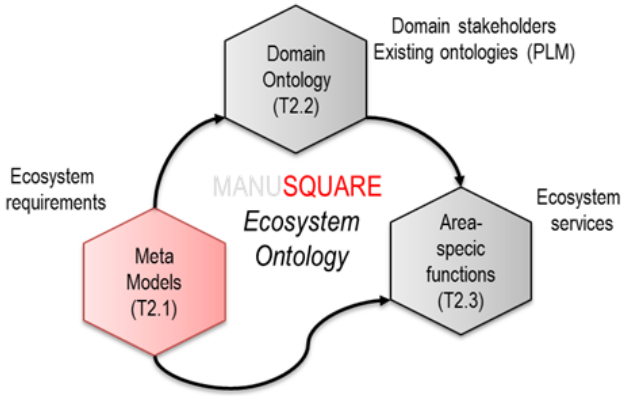


Figure 1 - WP2 tasks relationship

Moreover, it is expected that the data model will undergo many improvements according to the needs that will be identified during the platform tools development, as it is expected from the activities belonging to the Task 2.3 where the model will be extended in order to support the functioning of the service-providing applications developed in WP4. The same kind of improvement will be taken from the results coming from the Task 2.2 where the data model will be expanded by translating the domain-specific knowledge of the industrial sectors addressed in the project. Therefore, this document shall be considered as a live document subjected to a continuous review process aimed at keeping it up-to-date.

In order to better understand the results obtained within the Task 2.1, here reported, the following sections are meant (i) to describe the platform architecture and (ii) to summarize services and functionalities provided by the platform and identified during the activities belonging to the Task 1.2.

2.1 Overall System Architecture

This section is meant to sketch the overall system architecture of the MANU-SQUARE platform, describing the main involved components and their interactions.

The first level of definition of the system architecture (see Figure 2) depicted below, gives a short overview of the high-level components that constitute the system architecture with a description of their main responsibilities inside the MANU-SQUARE system and their relationships.

The software architecture underlying the MANU-SQUARE Marketplace Platform (MMP), shown in Figure 2, is meant to support the consistent and secure flow of information that, beginning with the data gathering in the companies' shop-floor, allows to aggregate real-time information to achieve a constantly updated snapshot of the ecosystem status, thus fostering the provision of high value-added services to its members.

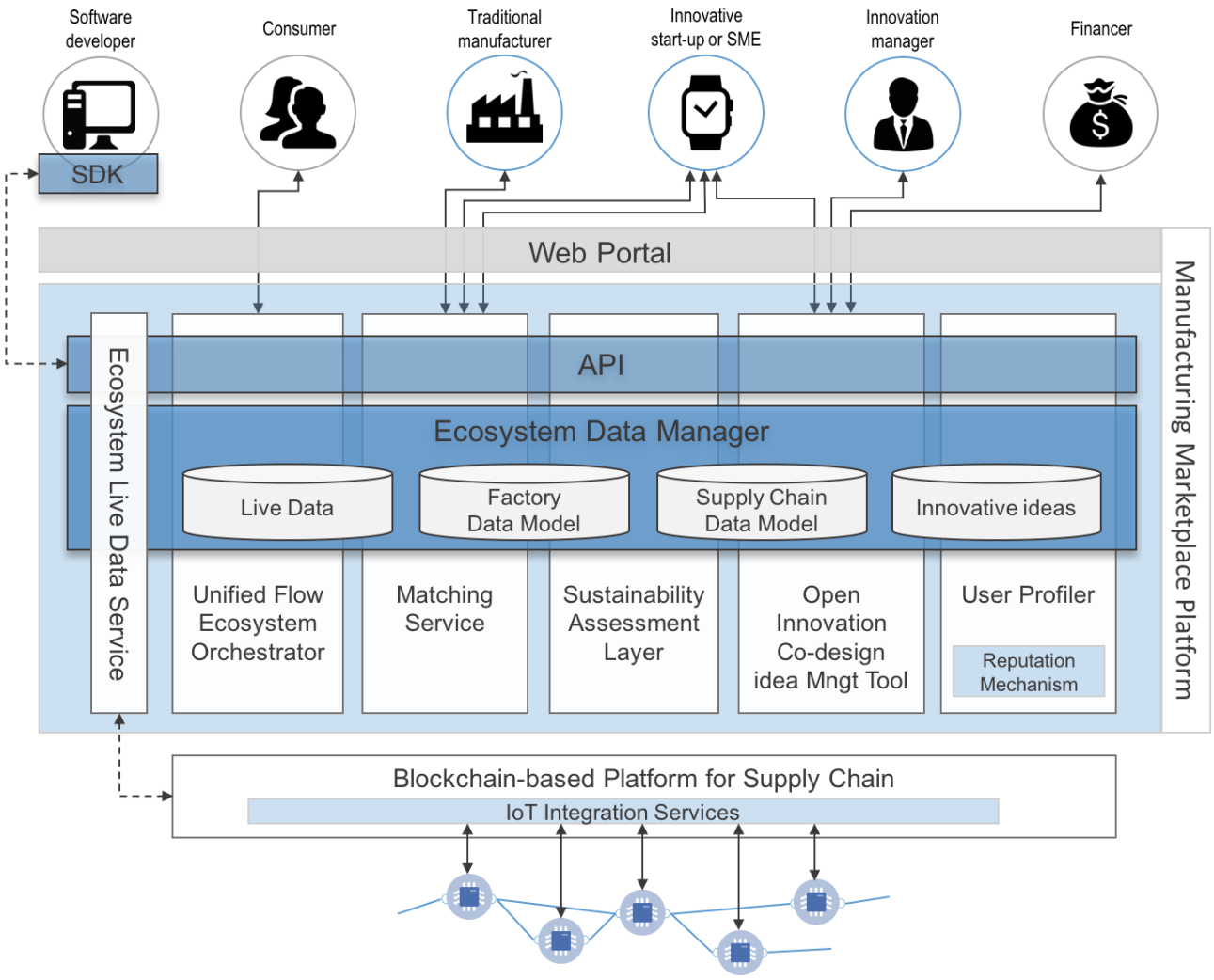


Figure 2 MANU-SQUARE platform architecture

The MMP functioning strictly relies on the role of the Blockchain-based Platform for Supply Chain (BPSC) that has a twofold aim: on the one hand, it manages the processes and related transactions at any node of the supply chains generated within the ecosystem while, on the other hand, it connects the IoT devices within the single company node to support automatic data gathering.

The MMP acts as the core managing system of the whole solution and with its components provides functions at three different levels: **data management**, **service provision** and **external integration**.

At **data management** level it has two main functions: (i) it exposes interfaces for data connection with the BPSC through the Ecosystem Live Data Services thus feeding the semantic representation of the ecosystem in the inputting direction while dictating actions such as user-driven supply chain configuration in the outputting direction; (ii) it keeps the various data repository (Live Data, Factory Data, Supply Chain Data, Innovative Ideas, etc.) updated through the Ecosystem Data Manager taking also care of the persistency of the data models.

At **service level,** it leverages on a set of modules that, by reasoning on the semantic representation of the data models, provides complementary services for the platform users (both human and other software modules). These service-providing modules act cooperatively making a cross usage of the functionalities that each module offers to provide higher added-value services. The main service-providing modules are:

- *Matching Service*, the module stands at the core of the demand-supply matching within the MANU-SQUARE marketplace. It uses non-aprioristic relations offered by the semantic nature of the data models to identify and rank all the possible and most fruitful connections between companies' needs and offers.

- *Unified Flow Ecosystem Orchestrator*, an optimiser that abandons the one-to-one approach followed by the Matching Service to adopt an ecosystem-level perspective meant to enhance efficiency in the use of the resources (materials, energy, wastes, etc.) flowing between the ecosystem members.
- *Sustainability Assessment Layer*, a tool that calculates LCI and LCIA to measure the environmental sustainability of the processes and flows taking place in the ecosystem with the aim to complement the ranking system offered by the Matching Service with particular lens that monitors sustainability impacts.
- *Open Innovation Co-Design Idea Management Tool*, a platform for creative user interaction where new projects can be posted and followed to incentivise other actors' involvement in the development of the idea in an open innovation approach.
- *User Profiler and Reputation Mechanism*, the management tool of all the users (companies but also physical persons like the Innovation Manager) that allows the characterisation of demand, supply, know-how, competences and all the other information needed by the other modules to carry out their reasoning.

Eventually, at the **external integration level**, the MMP provides means for user interaction, on the one hand, and for platform expansibility through integration of new service-delivering modules, on the other hand. To this end it relies on:

- *Web Portal*, a web-based application composed by a set of web pages which present a UI able to exploit the MMP functionalities passing through the MMP API.
- *API* that represents a set of REST web services that allow external actors to interact directly with the functionalities and services exposed by the MMP. This kind of integration can be exploited by whoever wants to integrate his/her own applications directly with the MMP (e.g. software houses proposing a new service module to complement the platform, companies wishing to integrate the platform services in their own web portal, etc.).

2.2 MANU-SQUARE functionalities and ecosystem description

In order to develop the MANU-SQUARE data model, a set of requirements for ontologies creation has been defined according with the work carried out in Task 1.2 where platform specifications have been depicted. These specifications represent a relevant starting point for the data model design and development, defining, first, the main elements it has to consider in order to describe the manufacturing ecosystem MANU-SQUARE relies on and, secondly, paving the way for the identification of non-trivial connections among resources shared across the platform.

The MANU-SQUARE data-model constitutes the backbone of the entire platform. For this reason, it is necessary to have a clear idea of which entities and attributes have to be included since the beginning of its development. This shall not preclude the possibility to add other elements, but starting from clear requirements, the data-model development and its update will be managed more efficiently, producing more effective results.

According with the project plan, Task 2.1 and Task 1.2 have been carried out in parallel. Nevertheless, the two teams responsible of the activities closely cooperated to produce consistent results, each one taking into account and exploiting the outputs of the other. For the data model design and development, the most relevant outputs coming from Task 1.2 are the MANU-SQUARE specifications definition framework, represented in Figure 3, and the use cases the platform is expected to support, summarised in Table 1.

The framework covers the main elements of the MANU-SQUARE ecosystem that have to be considered for data-model design and development, namely Stakeholders, Needs, Functionalities, Tools, Components and Services. However, for data model definition purpose, the main requirements that come from the MANU-SQUARE platform framework are related to the platform functionalities:

- **Production capacity, know-how capabilities and by-product matching**: the data model has to include entities and attributes able to describe production capacity, know-how and by-products in order to support the logics that will be included in the matching tool.
- **Sustainability Assessment:** one of the innovations introduced by the MANU-SQUARE platform, that is not included in other manufacturing sharing platforms, is the possibility to evaluate suppliers from a sustainability

point of view. The data model has to support the description of the sustainability performance of a resource enabling the supplier assessment.

- **Ecosystem optimization:** the platform is able to support the ecosystem optimization adopting for the Suppliers' ranking a specific objective function. This is developed in order to suggest the best solution to improve the overall sustainability and not only the one of the two involved companies (Customer and Supplier).
- **User Profile management:** each user who creates an account on the platform has different characteristics that have to be considered in the data model to allow the access and the use of the platform, but also to support search, matching and assessment.
- **Reputation management:** each user is assigned a reputation score that is created using the historical transactions and feedbacks from other users. The management of this information is fundamental to guarantee reliability and trust.
- **Suppliers assessment:** after the identification of the matching suppliers, the platform has to be able to rank them on the different assessment parameters (e.g. reputations, sustainability performances) considering their related weights/relevance set according to the customer's priorities.
- **Certifications management:** the platform allows Auditors and Regulators to certify players through a verified and secure certifications management system. Certifications awarded have to be securely stored and associated with the users' profile.
- **Trust management:** the functionality supports the management of information across the platform. Users who insert information and data are able to decide which level of accessibility provide to their information, deciding if sharing the with other users or only with the platform.
- **Communication support:** the platform supports communication among platform users, streamlining connections and mediating the interactions among parties.
- **Innovation management:** the platform supports the management of social interface where, starting from a defined content, different users can provide tracked and structured contributions. To this end, the platform administrates the introduction of new ideas, the gathering of feedbacks from potential users, the flow of contributions deriving from third parties and the creation of project related contributors' lists with public or private accesses. In this context, the data model's entities and attributes have a relevant role to facilitate and guarantee the ownership of contribution, and also to facilitate the search of projects, contributions and ideas.
- **RFQ management:** this functionality provides not only the infrastructure to enable the definition and management of quotations, but it also manages the level of visibility of the quotations and of the partners exchanging requests and transactions, according to business model guidelines**.**
- **Transactions management:** it supports the creation of traceable transactions across the platform value network.
- **Support to platform expansion:** this functionality supports the extensibility of the platform by managing the integration of third-party services or the access by third-party applications to the functionalities of the platform itself. The data model has to be designed and developed in order to facilitate the data utilization from a third-party application, providing the documentation and the neceesary support.

The data model has to be designed and developed to be able to support the provision of all these functionalities, taking in consideration the industrial context in which the MANU-SQUARE platform will be applied, characterized by industrial plants, machines, tools, operators, resources, materials; the different characteristics of the users who will use the platform (e.g. manufacturing companies, SMEs, start-ups), and the tools that will access and use the data.

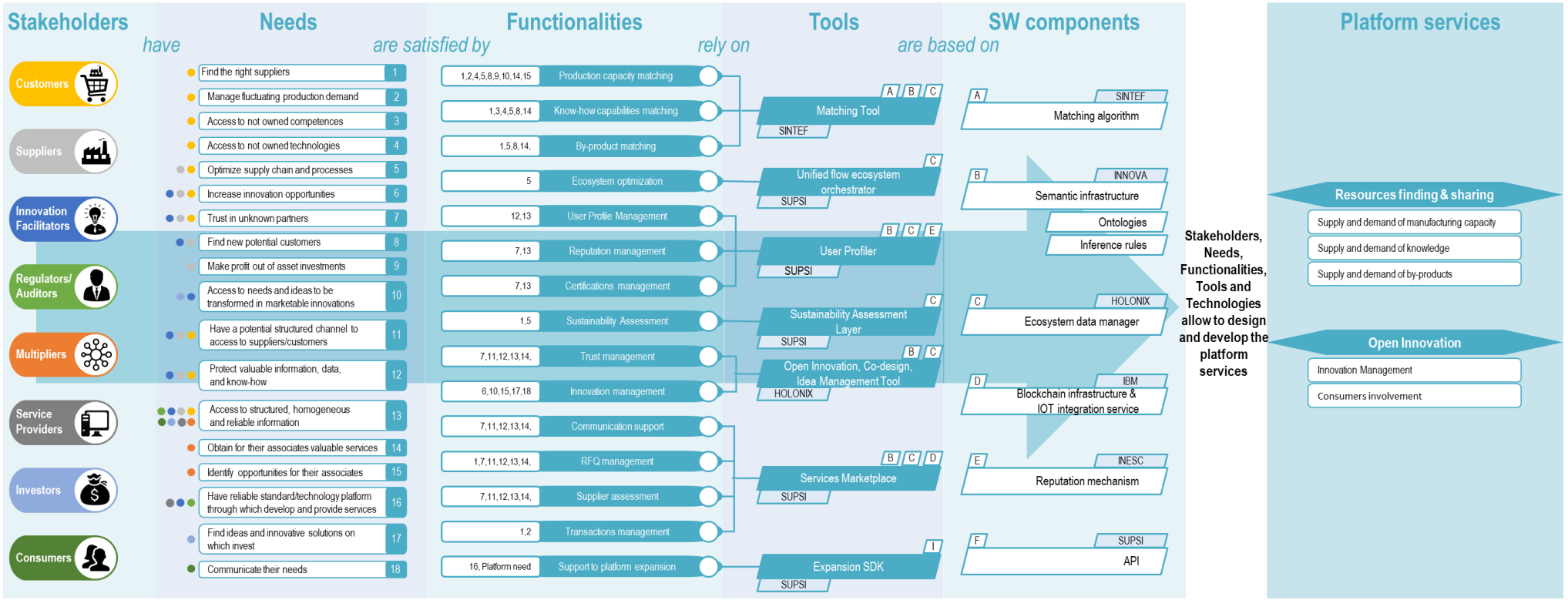


Figure 3 MANU-SQUARE platform framework for specifications definition

Beyond the functionalities supported by the MANU-SQUARE platform, an introduction to platform use cases, summarized in Table 1, allows to understand the main mechanisms and logics that will drive the use of platform services, supported by the data model designed in this task. The use cases have been developed during Task 1.2 and describe different platform's service utilization scenarios. The table contains the name of the use case, its description and the input data that could be provided by the users to correctly use the platform and the related service (for more details about the use cases, see Task 1.2). The input data are examples assumed by the use cases' authors. Task 1.3 will develop a more detailed analysis of the type of data that can be provided by the users in the project reference scenarios that will be used for data model instantiation in the specific project validation domains.

Use case name	Description	Input data
Resources finding and sharing – Supply and demand of manufacturing capacity – Customer side	Manufacturing company working in the wood sector is looking for a machine to produce 300 customized metal components to be integrated into its products. The company needs the manufactured parts in three weeks from the date of the request. They want to receive the RFQ answer in a week from the date of the request. The supplier must be ISO 9001 certified.	**Customer enters in the platform's search interface:** Customer location (Lugano), Component design, Material (steel), Size (10*20*30 cm), Quantity (300), Expected process to be used (Milling), Expected machine (CNC 5 axes), RFQ response time (1 week), Production lead time (3 week), Certifications (ISO 9001) **Suppliers have to describe their attributes and resources in order to allow the match.**
Resources finding and sharing – Supply and demand of manufacturing capacity – Supplier side	A company working in the metal sector purchased a CNC machine to produce a new product. Since this is not already well-known on the market, its demand is currently low. For this reason, the company aims to saturate the machine looking for external production orders.	**Supplier enters in the platform's dashboard:** supplier location (Lugano), products (cantilever roofs, intermediate floors, stairs, metallic carpentry), materials (steel, iron), sector (production machining & fabrication, automotive, plastics, defence, consumer), time availability (20 hours a week), resources (1 CNC machine with related features/characteristics description (e.g. machine name, brand, n° of axes, tools, working cube, etc.)) **Regulator/auditor certifies the Supplier.**
Resources finding and sharing – Supply and demand of knowledge – Customer side	A manufacturing company, named Woody SA, producing wood products for the luxury sector, has received a custom order from a new customer. Woody SA aims to satisfy customer's request in order to create a lasting relationship. The customer requires 10 ship wheels of precious wood which have to be covered with water-repellent resin. Woody SA usually produces luxury furniture which not require this kind of process. Therefore, it has not competences on the type of resin to use, how to apply it. For this reason, it aims to receive the support from an experienced supplier in order to be sure to satisfy customer expectations.	**Customer enters in the platform's search interface:** sector (wood, vessel-building, wood processing), product type (boat component, ship-wheels, water-repellent resin), process (resin application, resin covering), component size (120*120*30 cm), quantity (10), RFQ response time (2 week), production lead time (4 week), certifications (ISO 9001). **Suppliers have to describe their attributes and resources in order to allow the match.**
Resources finding and sharing - Supply and demand of by-product - Supplier side	Desia SA is a company settled near Lion, in France, manufacturing high-quality mirrors for aeronautic applications. The process Desia applies to polish their mirrors generates high-quality polishing sand that they cannot use two times due to quality reasons. They aim to find a company interested in buying the sand for	**Supplier enters in the platform's dashboard:** user location (Lion), sector (aerospace), product type (mirrors), by-products/wastes (sand), by-products/wastes characteristics (1,5 µm per grain, aluminum micro-spheres), quantity (100 kg/week), expected exploiting process (industrial sanding, sandblasting), expected sector of application (machine tools manufacturers),

	other applications.	expected refining process; in-house/third-party (dry-sifting; in-house) **Suppliers have to describe their attributes and resources in order to allow the match.**
Innovation Management - Open Innovation Management	A company working in the textile sector aims to innovate its curtains product lines, introducing a dedicated one for the smart-housing. It has already a conceptual idea, but it wants to develop a more advanced concept. To do this, it aims to involve a third-party player.	**Customer enters in the platform's search interface:** idea description (text), sector (home decor, textile, smart house, smart textile), product (curtains), innovative elements (curtains with integrated led), target contributors (designers, innovation managers, sector experts), innovation stage (Concept development) Innovation Facilitators have to describe **Suppliers have to describe their attributes and resources in order to allow the match.**
Innovation Management – Consumers gathering	A company working in the textile sector aims to innovate its curtains product lines, introducing a dedicated one for the smart-housing. It has already a conceptual idea, but it wants to evaluate the market interest in this solution.	**Customer enters in the platform's search interface:** Idea description

Table 1 Use-cases description from Task 1.2

3 STATE-THE ART IN DATA MODELLING

Data modelling is a complex process, consisting of defining and analysing data requirements needed to support the business processes of an organisation. The data requirements are recorded as a conceptual data model with associated data definitions. Data modelling defines the relationships between data elements and structures [DIG01]. Data modelling techniques are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The modelling of data, based on state-of-the-art data modelling languages is required for each planning or continuous adaptation of factories. The following types of modelling techniques are analysed in § 3.1 to § 3.5:

- Entity Relationship Modelling (ERM),
- Unified Modelling Language (UML),
- EXPRESS,
- Integration Definition (IDEF),
- Object Role Modelling (ORM).

3.1 Entity Relationship Modelling - ERM

An Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Among other Items, the Entity Relationship Modelling is a method to model databases in order to produce a conceptual schema or semantic data model of a system and its requirements by a top-down approach. Created diagrams are called entity-relationship diagrams, ER diagrams, or ERDs.

One of the most widely accepted models is the ER Model of [1], [2]. It deals with static structural properties such as entities, attributes and relationships. The model is easy to use and to understand. It clearly shows all types of concept abstractions, various relationships, mapping constraints and cardinalities. An entity may be defined as an Item which is recognised as being capable to exist independently and which can be uniquely identified [3]. An entity is an abstraction from the complexities of some domain [4]. Each entity has certain characteristics, known as attributes. Such entities are represented by boxes. An entity can be a physical object such as a building or a shoe for example, an event such as a shoe sale, or a concept such as a customer transaction.

A relationship is an association among entities and is represented by diamond-shaped symbols. The list of components and the characteristics of the relationships constitute the relationship type [3]. Relationships can be considered as verbs, linking two or more nouns. One example is the relationship between an employee and a department. Relationships are described by diamonds, which are connected by lines to each of the entities.

Objects are used as synonyms for entities and relationships. A Class is used a synonym for a set of entities or a set of relationships. Often, object-oriented models do not distinguish between basic objects and association objects. One of the important advantages of Entity Relationship Models is that identification is maintained due to the hierarchical definition of types [3].

Entity Relationship diagrams do not show single entities or single instances of relations. They show entity sets and relationship sets. For example, a particular shoe is an entity. The collection of all shoes in a database is called an entity set. A relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation. Certain cardinality constraints on relationship sets may be indicated as well.

3.2 Unified Modelling Language - UML

Unified Modelling Language (UML) is a standardised general-purpose modelling language for object-oriented requirements analysis, modelling, specification and documentation of data and programs. Compared to other modelling languages such as the Entity Relationship Modelling, the description of static aspects as well as dynamic aspects of a system is enabled by UML. In particular, UML supports the following features:

- aggregation of objects,

- allocation from functions to objects,
- interaction between objects and whole (business-) processes.

UML aims at modelling the whole spectrum from informal communication with operators up to program documentation. Therefore, a multiplicity of diagram types is provided. Thus, the different phases of a software-developing process are supported by different diagram types. UML defines nine different types of diagrams [5]:

- class diagrams,
- sequence diagrams,
- collaboration diagrams,
- object diagrams,
- state chart diagrams,
- activity diagrams,
- use case diagrams,
- components diagrams,
- eployment diagrams.

Use case diagrams are used to describe different application scenarios, which are created by the developer for the requirements analysis of the system. For the illustration of a structural and static system architecture, the diagram types component diagram and class diagram are used. For the representation of the dynamic behaviour of a system, UML provides several types of diagrams like activity diagrams, sequence diagrams and state chart diagrams. The class diagram is primarily used for data modelling.

3.3 EXPRESS

Data models formally define data objects and relationships between data objects. Typical applications of data models are the development of databases and the enabling of the data exchange for a particular area of interest [6].

EXPRESS is a standardised modelling language for product data and is formalised in the ISO Standard for the Exchange of Product model STEP (ISO 10303) [7]. EXPRESS is similar to programming languages such as PASCAL. Various data types can be defined within a schema together with structural constraints and algorithmic rules. Furthermore, EXPRESS enables the formal validation of a population of data types.

An EXPRESS data model can be defined textually as well as graphically. The textual representation within an ASCII file is important for a formal verification. The graphical representation is often more suitable for human use such as explanation and tutorials. The graphical representation, called EXPRESS-G, cannot represent all details that can be formulated in the textual form.

EXPRESS-G is a standard graphical notation for information models [8]. It is useful for displaying entity and type definitions, relationships and cardinality. This graphical notation supports a subset of the EXPRESS language. One of the advantages of EXPRESS-G over EXPRESS is the possibility to present the structure of a data model more understandably. However, complex constraints cannot be formally specified by EXPRESS-G.

3.4 Integration DEFinition - IDEF

IDEF (Integration DEFinition) [9] is a family of modelling languages in the field of software engineering. They cover a wide range of applications, from functional modelling to data modelling, simulation, object-oriented analysis and design as well as knowledge acquisition.

Sixteen methods, from IDEF0 to IDEF14 (including IDEF 1 as well as IDEF1X), are designed to capture a particular type of information through modelling processes. These methods are used to create and analyse graphical representations of various systems, as well as to support the transition between IDEF methods. The following IDEF methods have been defined:

- IDEF0 : Function modelling,

- IDEF1 : Information Modelling,
- IDEF1X : Data Modelling,
- IDEF2 : Simulation Model Design,
- IDEF3 : Process Description Capture,
- IDEF4 : Object-Oriented Design,
- IDEF5 : Ontology Description Capture,
- IDEF6 : Design Rationale Capture,
- IDEF7 : Information System Auditing,
- IDEF8 : User Interface Modelling,
- IDEF9 : Business Constraint Discovery,
- IDEF10 : Implementation Architecture Modelling,
- IDEF11 : Information Artefact Modelling,
- IDEF12 : Organisation Modelling,
- IDEF13 : Three Schema Mapping Design,
- IDEF14 : Network Design.

The most-widely recognised and used IDEF methods of the IDEF family are IDEF0, a functional modelling language, and IDEF1X, which addresses information models and database design issues.

IDEF0 is a method for modelling the decisions, actions, and activities of an organisation or system. IDEF0 was derived from a well-established graphical language, the Structured Analysis and Design Technique (SADT). The United States Air Force commissioned the developers of SADT to develop a function modelling method for analysing and communicating the functional perspective of a system. Effective IDEF0 models give support to organise the analysis of a system and to promote good communication between the analyst and the customer. IDEF0 is used for establishing the scope of a functional analysis. As a communication tool, IDEF0 enhances the involvement of different domain experts and a consensus decision-making through simplified graphical devices. As an analysis tool, IDEF0 assists the modeller in identifying what functions are performed, what is needed to perform those functions, what the current system does right, and what the current system does wrong. Thus, IDEF0 models are often created as one of the first tasks of a system development.

IDEF0 includes both a definition of a graphical modelling language (syntax and semantics) and a description of a comprehensive methodology for developing models.

For example, IDEF0 methods are used to model the functions of an enterprise, creating a graphical model that shows different functions, the used resources, the products, and all relationships among the functions.

3.5 Object Role Modelling - ORM

Object Role Modelling (ORM) is a fact-oriented method for creating a conceptual model. In Europe, the method is called Natural Language Information Analysis Method (NIAM), as well. The formal basis, on which business operations are defined, is provided by the fundamental database of a system. Therefore, the focus of ORM is on data modelling, although several ORM extensions have been intended for process or event modelling [10]. The function modules of Object Role Modelling are data objects (a data element or a set of elements) and their coherence. In ORM diagrams, these data objects are represented by ellipses and the relationships between such objects are represented by lines and subdivided boxes [11]. Object Role Modelling uses a natural language, as well as intuitive diagrams, which can be settled with different examples [10]. Entities and objects can be hardly described by ORM independently from relationships. The modelling language describes so called "facts", which are represented in terms of objects (entities or values). Unlike Entity Relationship Modelling (ERM) and Unified Modelling Language (UML) class diagrams, ORM does not use attributes in its basic models [11], but treats all facts as relationships (“roles”). Roles are titled by role names, displayed in square brackets. All fact structures are described as fact types and not as attributes. These fact types can be unary (e.g. Company produces), binary (e.g. Company was founded on Date), ternary (e.g. Factory was build in Country in Year) or higher [12]. ORM considers graphical representation as well as textual languages for modelling and

querying information at a conceptual level. Some tools also provide an automatic transformation between the graphical and textual representation[11]. Graphical language of ORM allows to detect and express constraints and to visually transform schemas into equivalent alternatives [12]. ORM also includes effective modelling procedures for constructing and validating models. ORM is able to extract the business data requirements from the domain experts into a conceptual model that can be easily understood and can be verified by those same domain experts [14].

Furthermore, a second generation of ORM notation (ORM 2) has been developed based on the ORM language. This notation is an open source plug-in to Microsoft Visual Studio.NET [11]. Similar to ORM, ORM 2 provides a graphical and textual notation.

The ORM 2 graphical notation is more compact. This leads to significant reduction of the diagram size. Furthermore, the English-specific symbols have been replaced by language-neutral symbols to improve localization in different language communities [13].

Unlike the first ORM solution, the ORM 2 textual notation can be used as an input for models, instead of being a pure output language for verbalizing diagrams. Furthermore, ORM 2 supports a formal, textual language for the declaration of ORM schemas and ORM queries. The generation of codes to implement the semantics that are conveyed by the textual notation, is supported by specific tools [13].

4 ECOSYSTEM DATA MODEL

This section documents the *Data Model* developed in the context of Task 2.1. It is organized into a General Model section (see Figure 4) with 5 sub-sections that correspond to as many main semantic areas of the data model:

1. ***Factory Model (§ 4.1)***: this section documents the set of classes needed to provide a description of a company, characterizing manufacturing processes and involved resources.
2. ***Stakeholder Model (§ 4.2)***: it introduces the concepts of Stakeholders (Supplier, Investor, Innovation Facilitators, Customers) who are the main user of the platform functionalities.
3. ***Innovation Ideas Model (§ 4.3):*** it specifies the way to organize the different ideas collected by the platform, linking them with their stakeholders.
4. ***RFQ Model (§ 4.4):*** this section presents a set of classes required to define the concept of Request for Quotation (RfQ).
5. ***Sustainability Assessment Model (§ 4.5)****:* this section contains all the classes needed to perform the evaluation of processes from a sustainability point of view.

Each section is introduced by a class diagram (based on UML Class Diagram) which contains only classes composing that specific data model area and their relationships with the main classes belonging to the other data model areas. Therefore, it is possible to find out the same class (e.g. *Stakeholder* class) in many different diagrams that is documented only once in the proper semantic section. The complete data model is depicted below (Figure 4) where the interconnections between each sub-section of the diagram are highlighted with coloured boxes.

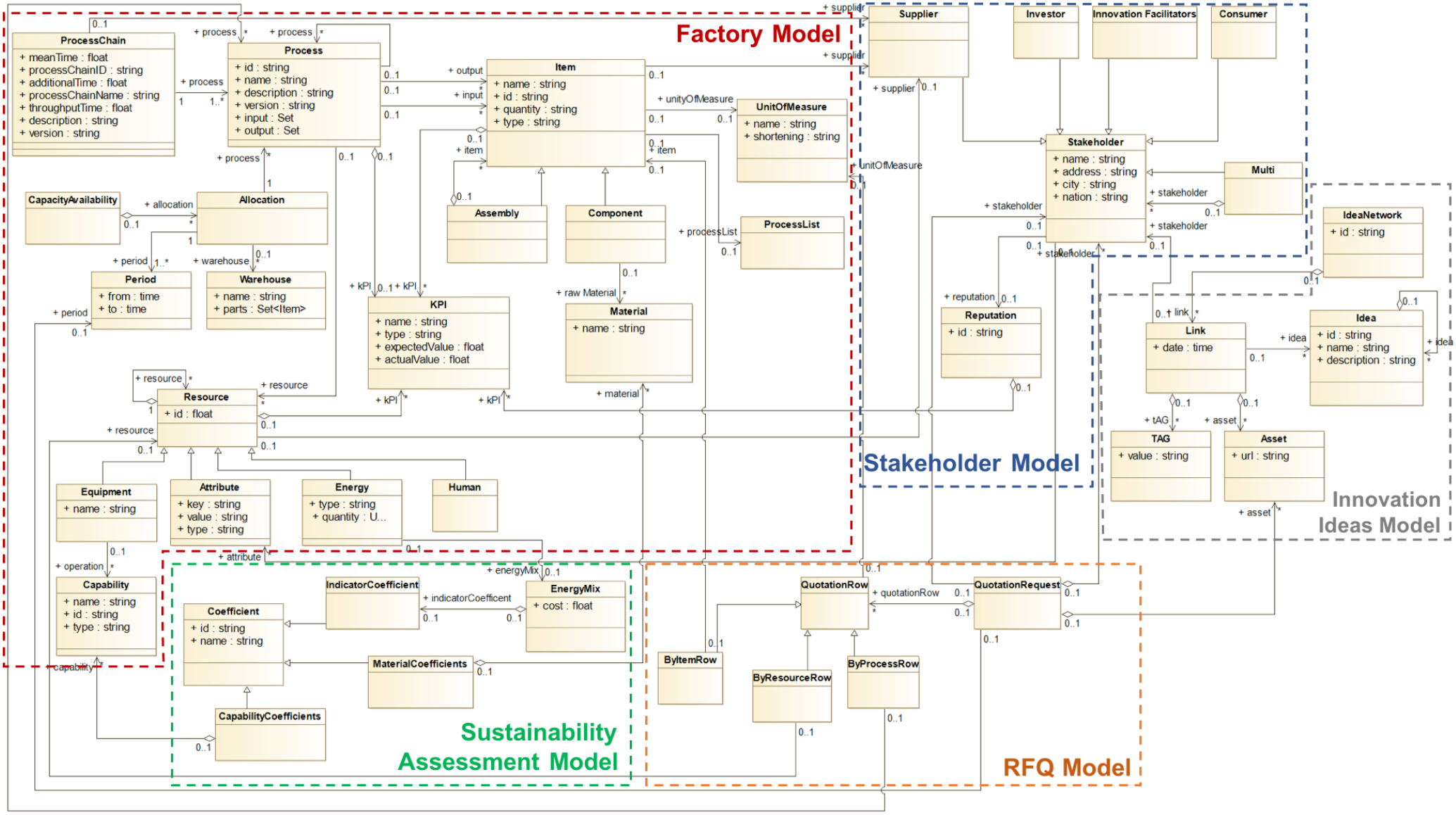


Figure 4 General Data Model Overview

Each class is then presented in a dedicated sub-section that is organized with:

- a short description of what the class represents, how it fits into the overall picture and which are its relevant generalization relationships, when applicable;
- a table, like Table 2, reporting all the fields of the class (represented in the UML diagram by attributes and relationships).

Whenever relevant for improving the comprehension of the model application, sample instance diagrams (based on UML Object Diagram notation) are provided. This document represents a first general version of the data models necessary for the functioning of the platform. A more specific version is assigned to Task 2.3 where both area of functionality and application domain will be contextualised.

Field	*Type*	*Description*
Name of the attribute or relationship	Field type. Collections are identified by the 2 square brackets "[ ]"	Short description of the field with indications on default values and value constraints.

Table 2 Field documentation

Parts of the data model, such the Factory description, may appear in a more detailed version than others. The different level of detail is motivated by the need to develop a model general enough to enable scalability and extensibility in the following sector-related implementation, but still able to provide a sufficient level of detail enabling concepts disambiguation.

The class attributes that have been reported in this document support the class documentation, but they can be modified as desired in the following phases of the project.

4.1 Factory Data Model

This subsection describes the most relevant aspects of a factory. The central aspect of this diagram, around which the other aspects have been designed, is the process. The process has been described taking the cue from the IDEF0 modelling standard. For this reason, the parallelism between Figure 5 and the main sections of the class diagram shown in Figure 6 is evident.

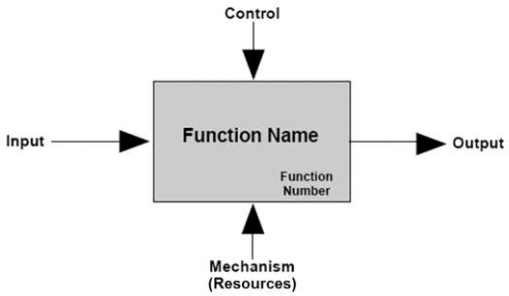


Figure 5 IDEF0 Building blocks

The choice of names has been kept as close as possible to the IDEF0[1] notation, in order to facilitate the communication between the various actors involved in the development of the platform.

The *Process* class has input and output relationships with class *Item* that represents a specific level of a BOM. These relationships map the input and output defined by the standard. The "Control" part is instead modelled with the sub diagram represented by the main *Allocation* and *CapacityAvailability* classes. The last component of the standard is the "Resources" that are grouped in the diagram under the *Resource* class.

[1] IDEF: www.idef.com/idefo-function_modeling_method/

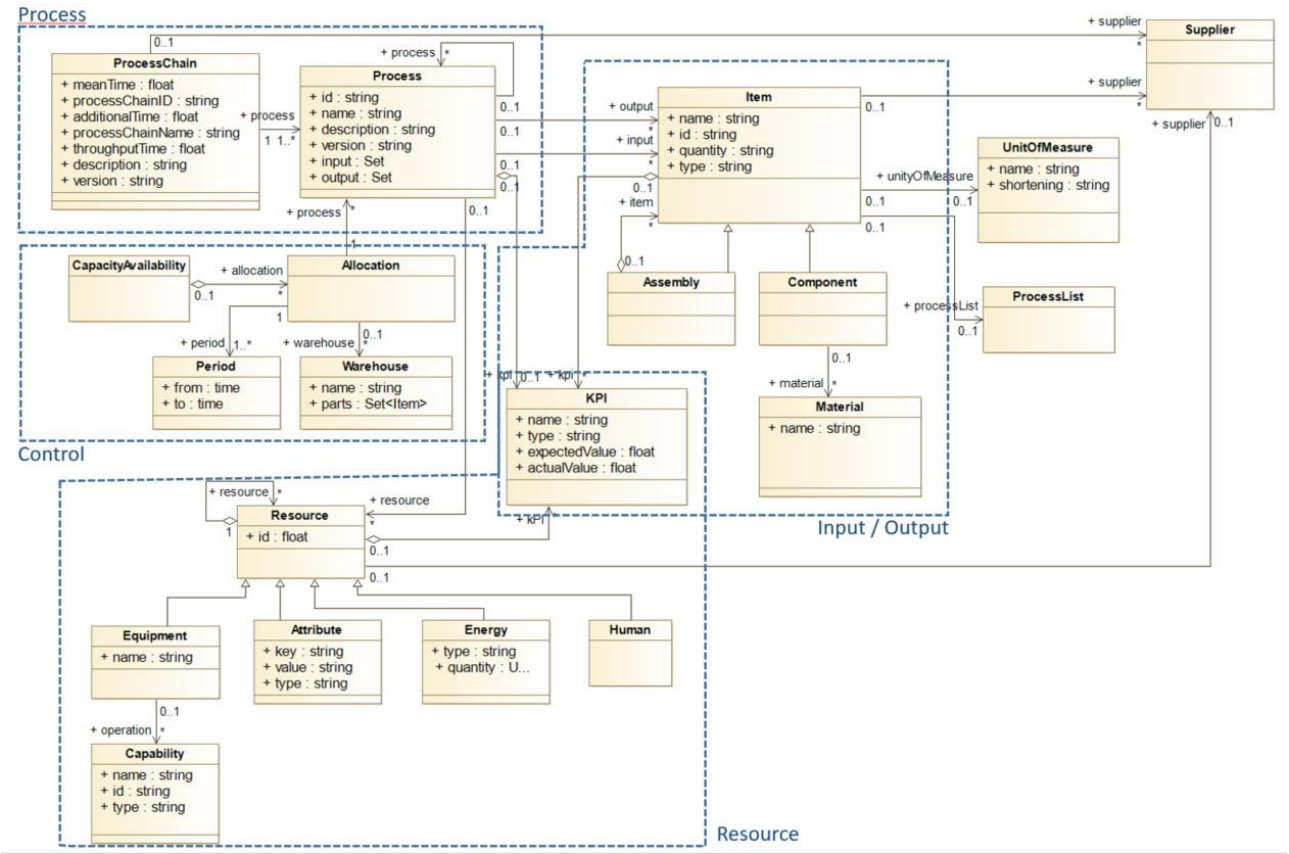


Figure 6 Class Diagram of the Factory model

In the following section, all the classes are described in detail. In particular, after a brief description of the single class, all the relationships with the other classes are described. At the end of each subsection, a table containing the description of each attribute belonging to the class is reported.

The Supplier class, for instance, is reported also in the diagram but it is described in the § 4.2.4. It is included also in this part in order to show the semantic relationship that links the two diagrams.

4.1.1 Process

Process represents the more significant class in the factory description. It aggregates the materials/sub-products/products/parts with the resources necessary to produce them. An example of process could be the milling process that transforms a work piece removing part of its material. The class *Process* has the following relationships with other classes of the diagram:

- **Input**: it connects *Process* with *Item*, that describes a Bill Of Material of a product (or part of it). It represents all the parts of a product that the process needs to use. For example, an Assembling Process has all the subcomponents that need to be assembled as input.
- **Output**: it connects *Process* with *Item* and represents all the parts that are produced as result of the process including waste materials, sub products and by-products. As described in § 4.1.10, the various item types can be distinguished by field type of the Item class.
- **Resource**: it connects *Process* with *Resources* and represents all the resources needed to complete the process.
- **Process**: it connects *Process* with itself and represents the possibility to model a process as an aggregation of sub processes. This link permits to both specialize and generalize an existing process.
- **KPI**: this link permits to describe the relationship between *Process* and the *KPI*'s; it also allows the definition of a set of measures to assess the process.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	Name of the element, a more human readable tag that can be used to identify the element within the model.
description	String	Short text field to be used to shortly refer to the element.
version	String	Defines the version of the element.

Table 3 Process attributes

4.1.2 Process Chain

The combination of processes forms the so-called process chain. The scope of this class is to represent a sequence of processes meant to enable the description of complete manufacturing lines. For instance, an Assembly Line typically consists of workstations where operations are performed in order to transform the input item into the desired output. The tasks performed in the various assembly stations can be modelled as a Process Chain. While the *ProcessChain* class gives an overview of processes in breadth, the Process class, with its compositions, details the process in depth.

The class *ProcessChain* contains the following relationship:

- **Supplier**: represents the owner where the production line is implemented
- **Process**: contains the list of all the processes involved in the production system.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
meanTime	Datetime	Describes the average time between failures of hardware modules.
additonalTime	Datetime	Is the production downtime due to failures and additional processes
executionTime	Datetime	Consists of the set-up time and the runtime
description	String	A description of the production line
version	String	Defines the version of the element.

Table 4 Process Chain attributes

4.1.3 Resource

The resource entity has been formalized in order to express all the assets needed for the transformation of an input, such as material/sub-product, into an output. There are many types of possible resources and, at the moment, the model does not formalize all of them. Some of them have been specified in order to give a more precise idea of the Resource concept. The following relationships have been identified for the *Resources* class:

- **KPI**: the class *Resource* contains a list of KPI's which represents a set of measured performances related to the use of resources.
- **Supplier**: *Resource* is also associated with the class Supplier, which is the owner of the resource. For example, it is possible to have a list of all the enterprises that own a certain type of resource (for instance a machine type).
- **Resource**: it's a relationship that permits to aggregate basic resources into more complex resources. The composition allows to aggregate different resources, showing them to its own client (e.g. Process) as a unique resource. For example, equipment that needs the intervention of human operators could be an aggregate resource of equipment and human resources.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
kpi	Set<KPI>	Contains all the KPI's that monitor the resource
supplier	Supplier	Contains the owner of the resource
resources	Resource	Other available resources

Table 5 Resource attributes

4.1.4 Attribute

This class has been formalized in order to allow the definition, in a dynamic way, of more detailed information related to a certain resource. It extends the generic class *Resource* in order to take advantage of the aggregation mechanism offered by the class *Resources.* For example, the skill of a Human could be represented as Attribute.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
key	String	An identifier that describes the meaning of the attribute
value	String	The value that describes the value of the key
type	String	It's like a label in order to organize the instances of this class

Table 6 Attribute attributes

4.1.5 Energy

This class expresses the concept of energy consumption which can be directly associated to a process. This class responds to the need of having a generic description of the process' consumptions. It is also possible to aggregate other resources to express a more detailed point of view related for example to the consumption of a particular equipment.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
unit	UnitOfMeasure	The unit of measure that characterizes the energy consumption quantity
quantity	String	The value of the energy consumption

Table 7 Energy attributes

4.1.6 Human

A classic example of an industrial resource is a human resource represented by the *Human* class. Moreover, Human is a resource which can be typically composed by a set of *Attributes* in order to express its skills. It could be indeed useful to list the skills necessary for the execution of a process or related to an *Equipment*.

4.1.7 Equipment

Equipment entity models the generic concept of a resource capable of performing a *Capability*. This could be, for example, a metal cutting machine, as well as a tool (e.g. a drill) used by a human operator. The model maintains a high level of abstraction, so as not to risk to specialize it too much, making it useless in different industrial contexts.

Field	Type	Description
id	String	Identification of the element.

		The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
description	String	A description of the equipment
capability	Capability	What the equipment is capable of doing

Table 8 Equipment attributes

4.1.8 Capability

Capability is the description of the ability to act on a process input.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Name	String	A name that describes the action e.g. MILLING
Type	String	It represents the type of the capability (i.e. milling, drilling, cutting, sawing, etc.).

Table 9 Capability attributes

4.1.9 KPI

Such entity is necessary to define indicators in order to supervise and compare different factory aspects. Thanks to this entity, three kinds of indicators can be instantiated: resources Indicators, process Indicators and product Indicators.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Type	String	It's like a label in order to organize the instances of this class.
expected value	String	Represent the expected value of the indicator.
actual value	String	Represent the actual value of the indicator.

Table 10 KPI attributes

4.1.10 Item

This entity is meant to represent, in a generic way, any input or output of a certain process such as material, waste material, component, assembled parts, by-product, finished parts or the final product. In order to specify the different nature of an item, there is the type field.

Item is related to other classes using the following relationship:

- **KPI**: the class *Item* contains an aggregation of KPI's that represents a set of performance measures related to the use of it.
- **Supplier**: *Item* is also associated with the class Supplier, which is the owner of the Item. For example, it is possible to have a list of all the enterprises that possess a certain type of product (for instance a screw).
- **Process list**: *Item* is associated with *ProcessList*. This last class provide a list of all the operations that have been made on the particular item. Describing the operations that an item has undergone during its life cycle is particularly useful for sustainability analysis (e.g. LCA). These operations can be reported with consolidated values.
- **Measurement**: *Item* is also associated with *UnitOfMeasure*, in order to define the kind of the defined quantity.

Field	*Type*	*Description*
id	String	Identification of the element. The element is identified across the entire plant by this

		univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	A name that describes the Item
quantity	String	Represents the amount of Material contained by the Component
type	String	Allows to classify the item type
unitOfMeasure	UnitOfMeasure	Define the unit of the defined quantity

Table 11 Item attributes

4.1.11 Process List

This class summarizes all the operations performed on an *Item* in its history. It summarizes the list of processes carried out, including those outside the company, in order to have a centralized knowledge point necessary, for instance, for sustainability assessments (LCA). Extraction operations of a material can be expressed in this class.

Field	*Type*	*Description*
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.

Table 12 Process List attributes

4.1.12 Component

Component is an *Item* specialization that wants to represent the concept of a leaf in a BoM structure. A *Component* can be used like a material (such steel) or a very simple component (like a screw). Component has a relationship with the class *Material*:

- **Material**: *Component* can be associated with a *Material* specifying its structural nature.

Field	*Type*	*Description*
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
material	Material	The component composition

Table 13 Component attributes

4.1.13 Material

This class represents the way to define the chemical/physical structure of a component. It is possible to model simple materials such as iron, alloys such as brass or even more complicated materials for which a more detailed description is not needed.

Field	*Type*	*Description*
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	A name that describe the Material
type	String	This attribute is meant to define the type of the material such as raw or waste material.

Table 14 Material attributes

4.1.14 Assembly

All parts of an item that are not components can be represented as *Assembly*. *Assembly* is an aggregation of components. Thanks to this structure it is possible to model concepts such as semi-finished or assembled products.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Items	Set<Item>	The parts of the assembly

Table 15 Assembly attributes

4.1.15 Unit of Measure

This is a generic class to support the representation of units of measure. It is used to express how to interpret quantities.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	A name that describe the unit of measure
shortening	String	The abbreviation of the unit of measure

Table 16 Unit of Measure attributes

4.1.16 Capacity Availability

This class has been modelled to support the management of processes, and therefore of resources and items, in such a way a general vision of the availability of a certain resource implicated to a certain process is provided. It contains an aggregation of allocations using the following relationship:

- **Allocation**: this relationship allows to answer all the questions related to capacity sharing needs that have an expiration time.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.

Table 17 Capacity Availability attributes

4.1.17 Allocation

The allocation entity represents the scheduling of a process in a period given the availability of items necessary for the process. For example, a process or a resource (like an equipment) can be allocated in a defined period. In this way, the needs of a resource can be scheduled avoiding overlaps on the same resource. In this way, it is possible to have a planning of the main factory' components. This class has the following relationships:

- **Process**: *Allocation* is associated with the class *Process* in order to specify what should be allocated.
- **Period**: The link with *Period* permits to reserve a time slot.
- **Warehouse**: *Warehouse* entity is meant to describe the availability of all the items necessary for the process.

In case new types of allocation are needed there is the possibility to formalise a link between resource and allocation so to define the allocation of a resource independently from the process of transformation. According to the work progress, these issues will be clarified and the data model updated accordingly.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.

Table 18 Allocation attributes

4.1.18 Period

Period is characterized by a start date (from) and an end date (to) when, for instance, a process can be allocated.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
from	datetime	Start of the period
to	datetime	End of the period

Table 19 Period attributes

4.1.19 Warehouse

The warehouse is the concept that allows to model the presence in the factory of the necessary items necessary as input for the allocated process. In this way, the item can be booked if already in stock or ordered so that it can be available by the start date of the period.

Field	Type	Description
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	A name that describes the warehouse
items	Set<Part>	The items present in the warehouse

Table 20 Warehouse attributes

4.2 Stakeholder Data Model

This part of the data model represents the stakeholders who are involved in the platform use. In addition to describing their specifications, the reputation mechanism has been also modelled, with the possibility of defining different assessment KPIs depending on the context in which the stakeholder operates.

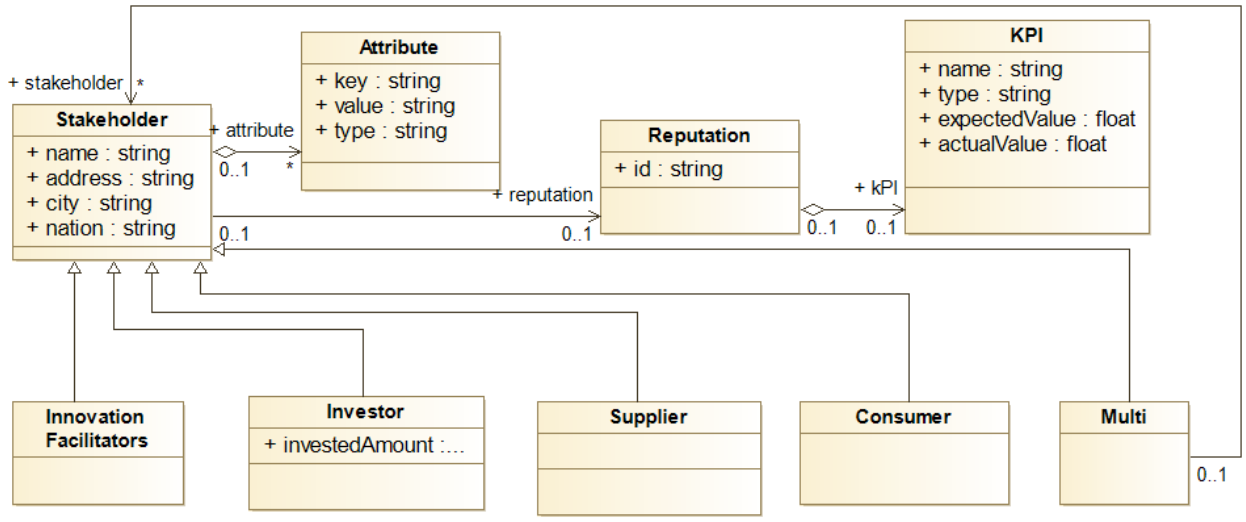


Figure 7 Class Diagram of the Stakeholder Model

A mechanism for the definition of multiple stakeholders has been provided in order to permit the combination of different roles. In the next section, each class is described with the related attributes.

4.2.1 Stakeholder

It is the class meant to abstract the concept of the platform' user. It is a very general concept and is used in many ways within the various data models. For example, the *Stakeholder* collects information about the reputation of the corresponding actor. In other cases, as for the sustainability assessment, stakeholder data can be used to estimate impacts related to the transport of components between different suppliers. As shown in Figure 7 this class has two relationships:

- **Attribute**: this link provides a way to enrich the description of the stakeholder depending on the context in which it is defined.
- **Reputation**: the link with the class *Reputation* contains all the data about the Stakeholder feedback.

Field	*Type*	*Description*
id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
name	String	A reference that can identify the represented subject
address	String	All the information about stakeholder's contacts
city	String	
nation	String	

Table 21 Stakeholder attributes

4.2.2 Innovation Facilitators

This class specializes the *Stakeholder* class. The concept behind such entity is to represent users, like universities or research centres, able to offer skills linked to technological developments, or users able to provide a professional network.

4.2.3 Investor

Investor specialized the *Stakeholder* class. The *Investor* represents the concept of a company or a person looking for new business and investment ideas.

4.2.4 Supplier

Supplier specialized the *Stakeholder* class. It can have resources that are not fully used and/or production capacities, capabilities, know-how and by-products that it aims to provide as a service.

4.2.5 Consumer

Consumer specialized the *Stakeholder* class. This player can look for production capacities to quickly ramp up production during peak periods, find production capacities, capabilities, know-how and by-products supplied as a service or third-party support for product/process innovation.

4.2.6 Multi

Since, in practice, the role of a stakeholder could be more complex than the one represented in this data model, this class is intended to offer the possibility of representing a stakeholder as an aggregation of stakeholders.

4.2.7 Reputation

Reputation is the class that collects all the evaluations that stakeholders have received from the collaboration with other stakeholders of the platform. This class foresees the following relationship:

- **Evaluation**: the class *Item* contains an aggregation of KPIs that represent a set of performance measures related to the use of it.

4.2.8 KPI

This class is already described in § 4.1.9.

4.3 Innovation Ideas Data Model

This part of the data model concerns the definition and the collection of innovative ideas. The formalization of this part starts from the *Idea* class that is modelled to allow the association with multiple contents and the addition of some labels that can be useful during the research phase of the idea. Ideas are also linked with stakeholders who conceived or are interested in them. Finally, *IdeaNetwork* entity has been formalized in order to trace all types of links created along the life of an idea. For example, once an idea has been created, the link "ideation" will be added to the *IdeaNetwork* and, as soon as it will be funded, the new link "funded" will be added.

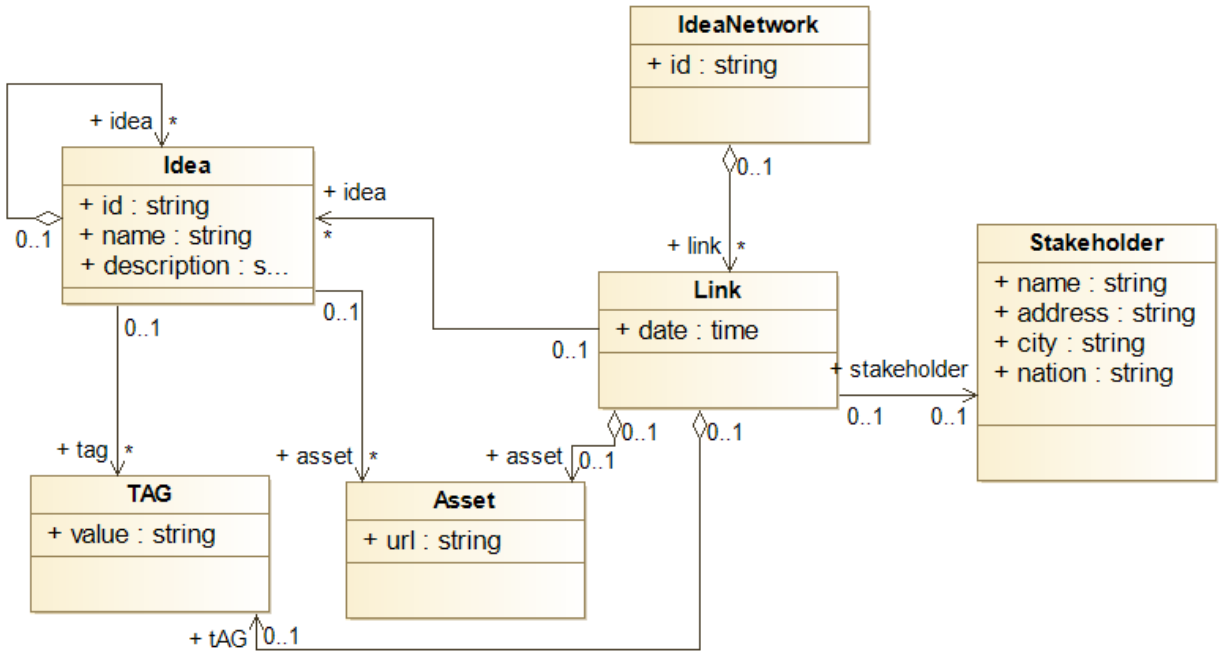


Figure 8 Class Diagram of the Innovation Idea

4.3.1 Idea

The concept of *Idea* has been modelled as a container of descriptions and external content classified through the use of labels. Thanks to the provided formalism, the aggregation of more ideas promotes its extensibility, in order to technically enable the constant evolution of the creative process. This class has the following relationships:

- **Tag**: Idea is linked to the class named "TAG" which is a keyword connected with the content of the idea. In this way, semantic research is enabled.
- **Asset**: the aggregation with the class *Asset* provides *Idea* with all the contents that can define it. Possible content could be: blueprints, schemas, diagrams, images, documents etc.
- **Idea**: an *Idea* can be represented as an aggregation of other Ideas. This allows at any time to extend the representation of the idea.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Name	String	Name of the element, a more human readable tag that can be used to identify the element within the model.
Description	String	Short text field used to describe the element.

Table 22 Idea attributes

4.3.2 TAG

This class represents the concept of tag ideas with keywords. It can be used in searches to improve the accuracy of the result.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Label	String	Represent the key word that characterizes the idea

Table 23 TAG attributes

4.3.3 Asset

This concept allows the data model to include additional documentation, of any nature, with the only constraint that it must be reachable using an URL. For example, the meeting notes, presentations, photos, CAD projects could be attached as an external information source.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
url	String	Reference to a resource in a computer network (file, HTTP, etc.)

Table 24 Asset attributes

4.3.4 Stakeholder

This class is already described in § 4.2.1.

4.3.5 Link

Link is the connection between an idea and a stakeholder. This connection is meant to encapsulate any kind of connection between these two classes through a Mediator Pattern[2]. In this way, connections between an idea and many stakeholders can be represented by simply instantiating *Link* several times (and vice versa).

4.3.6 IdeaNetwork

This is the grouping of multiple links to the IdeaNetwork. This entity has been formalized in order to trace all types of links created along the life of the idea. For example, once an idea has been created, the link “ideation” will be added to the IdeaNetwork and as soon as it will be funded, the new link “funded” will be added.

4.4 Request For Quotation (RFQ) Data Model

The MANU-SQUARE platform gives customers the possibility of looking for suppliers meeting their needs in terms of technologies, products, input materials, etc. The customer is also offered support for the management of the request for quotation thanks to the formalization of the RFQ data model (see Figure 9). The model is structured in order to reflect a product/service quotation including into rows. For each row, there is the option to request the cost to use a resource, to produce an Item or to perform a complete process. It is also possible to enrich the request by attaching additional information like CAD files, blueprints, descriptions etc. in order to provide the supplier with all the details needed for the quotation. The request for quotation is not limited only to a single supplier but can be addressed to many suppliers, allowing in this way the comparison between different alternatives.

[2] ISBN-13: 978-0201633610

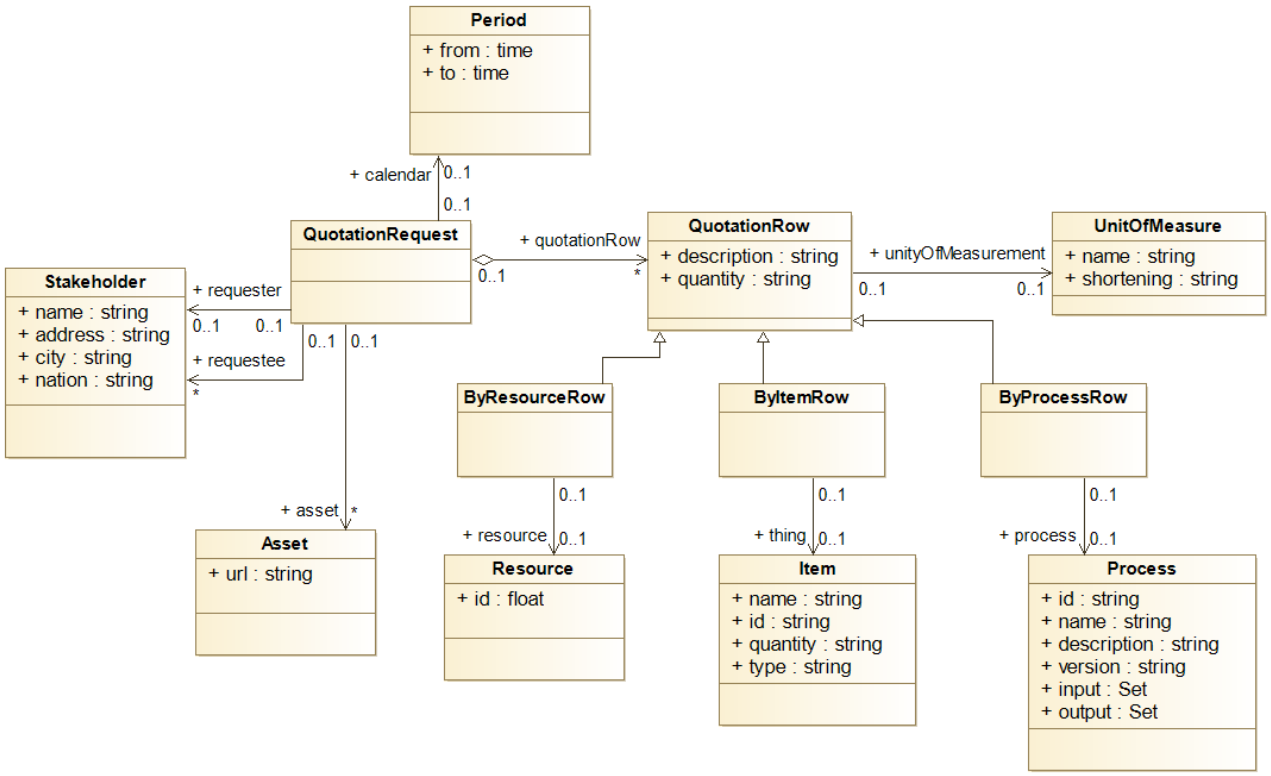


Figure 9 Class Diagram of Request For Quotation

4.4.1 QuotationRequest

A quotation request can be modelled as a document that contains a list of budget items. This class has the following relationships:

- **Requester**: a link between *QuotationRequest* and *Stakeholder* representing who made the request (the customer).
- **Requestee**: the other link between *QuotationRequest* and *Stakeholder* representing the addressees of the request (the suppliers).
- **Asset:** *QuotationRequest* can be enriched by adding contents defined as *Asset*. Possible content could be: blueprints, schemas, diagrams, images, documents etc.
- **Calendar:** the validation of the quotation request is time-limited.

4.4.2 QuotationRow

A budget item can be expressed through instances of this class. This is a superclass, currently with three specializations: ByResourceRow, ByItemRow and ByProcessRow. Such three different types can represent a quotation row. This specialization allows to have considerable flexibility in the type of requests sent to suppliers.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.
Description	String	Description for the supplier
Quantity	String	Amount of the desired object

Table 25 Quotation Row attributes

4.4.3 UnitOfMeasure

This class is already described in § 4.1.15.

4.4.4 Period

This class is already described in § 4.1.18.

4.4.5 Stakeholder

This class is already described in § 4.2.1.

4.5 Sustainability Assessment Data Model

The result of an available capacity search could return a very long list of suppliers. So how to choose between the results? A criterion could be the evaluation of the sustainability impact associated to a Supplier's Resources, Processes or Items. This section of the Data Model is therefore dedicated to the integration of Sustainability Related analysis into the MANU-SQUARE ecosystem.

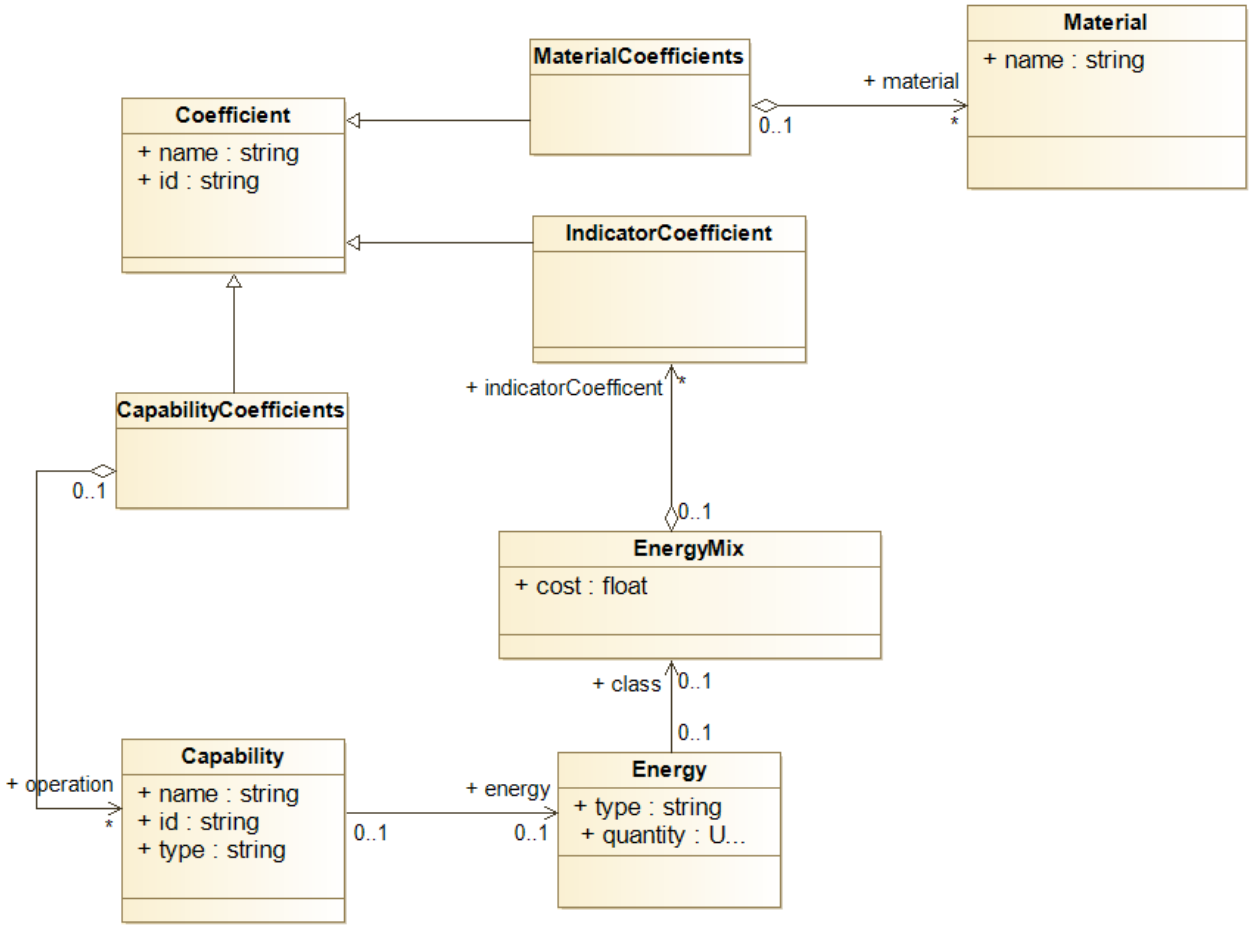


Figure 10 Class Diagram of Sustainability Assessment

4.5.1 Coefficient

Generic impact coefficient (therefore intended to be multiplied by a scalar value), represented by a N x M matrix where rows are indicators and columns life-cycle phases.

Field	*Type*	*Description*
Id	String	Identification of the element. The element is identified across the entire plant by this univocal attribute. Often, a Universally Unique Identifier (UUID) is used.

Table 26 Coefficient attributes

4.5.2 OperationCoefficient

Impact coefficient due to the execution of a given operation, it is multiplied by how much the operation is performed expressed in the operation characteristic value (example: cm^3 removed for a milling operation). The sustainability impact due to energy consumption are not considered here because they are using the EnergyMix coefficient.

4.5.3 Capability

This class is already described in § 4.1.8

4.5.4 Energy

Class representing a quantity of energy consumption.

4.5.5 EnergyMix

Impact coefficient due to the generation of electrical energy which depends on its means of production. It is multiplied by the energy spent in the execution of a given operation.

4.5.6 IndicatorCoefficient

Impact coefficient for indicators that does not follow a mathematical linear behaviour (namely social and economic one), its usage is defined on a case-by-case basis.

4.5.7 MaterialCoefficient

Impact coefficient due to a specific material extraction and refinement, it is multiplied by the quantity of a given material in each component.

5 CHOSEN TECHNOLOGIES AND CORE ONTOLOGY IMPLEMENTATION

This section describes an ontological representation of the MANU-SQUARE core conceptual model introduced in § 4. First, § 5.1 provides a short overview of ontology languages that can be used to implement and formalize the MANU-SQUARE core conceptual model into a formal, machine-interpretable ontology.

Then, § 5.2, provides a brief analysis of previous efforts in development of ontologies for manufacturing sector, with a special focus on their decisions about the ontology representation language/implementation choices. The purpose of this analysis was to understand the state-of-the-art approaches for the formal representation of manufacturing-sector ontologies.

§ 5.3 introduces the main requirements that have driven our decision process about an ontology representation language for MANU-SQUARE ontologies. Finally, § 5.4 outlines the main implementation aspects of MANU-SQUARE Core Ontology. The MANU-SQUARE Core Ontology is provided in Appendix A of this deliverable and will also be published on the web page of the project.

5.1 Ontology Languages Overview

First of all, the UML class diagrams are not ontologies, but the graphical representations of ontology concepts (classes).

The UML language offers a convenient way to capture the classes (or concepts) of data in a domain and the graphical representation of the domain classes, accompanied with textual descriptions of the classes, provides an excellent basis for discussing, negotiating and finally formalizing the agreed meaning (semantics) of the domain classes. Hence, the conceptual models of MANU-SQUARE Ecosystem data introduced in the previous sections as UML models will be translated into their formal and machine-interpretable ontological representation (see § 5.4 for details).

Also, it has to be clarified that the conceptual models of MANU-SQUARE Ecosystem data provided as UML models in § 4 should not be mistakenly understood as any of traditional data models (e.g. E-R models, relational database schema). The provided UML models are meant to be at the conceptual level, without any implementation-level details. There is a clear difference between a conceptual model (ontology) and a traditional data model. This project aims to build ontologies because of their advantages over the traditional data systems that are a relational database.

Traditional data models, in general, describe the data schemas and do not formally describe the meaning and semantics of the data. When someone builds a system using a traditional data model (relational database) approach, usually there is a Closed World Assumption (CWA). CWA is the assumption that what is not known to be true must be false. With the data model (data schema) implementation under CWA:

- you can only enter data that you know to be valid/true
- you are “encouraged” to enter complete information.
- there are no other data (what is not known to be true must be false)
- the data model implementation reveals no semantics about the data, but rather provide a data entry templates (e.g. relational database tables are templates to enter the valid and complete data).
- domain assumptions are not explicit
- because of the lack of formalized semantics of concepts and their instances (data), the knowledge and data sharing, integration and reasoning are limited.

On the contrary, a formalized conceptual model, or ontology, is a formal explicit description of concepts (classes, called also concepts) in a domain of discourse, properties of each concept describing various features and attributes of the concepts, and restrictions on properties [15]. 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 [15].

An ontology assumes the Open World Assumption (OWA), which is opposite than CWA. OWA is the assumption that what is not known to be true is simply unknown. Hence, with an ontology and ontology-based knowledge bases:

- you can enter what you know to be true.
- you can enter incomplete information/knowledge
- you (and the ontology applications) can understand the meaning of data and the applications can infer additional information/knowledge based on the ontology entailment and reasoning/inference rules
- ontology classes are simply sets of things, with precise semantics and do not act as a data entry templates in the system
- domain assumptions are explicit
- ontology boost knowledge or data sharing, integration and reasoning.

Ontologies, as a formal and explicit description of concepts in a domain, are implemented using ontology languages. The ontology languages are always formal languages, with concrete syntax and semantics. They are used to construct and validate ontologies, supported by the ontology development tools. Ontology languages, thus, allow the formal representation and encoding of domain concepts (e.g. foundational, industry or application specific domains), formalization and axiomatization of the domain concepts, and often include reasoning rules that support the knowledge processing and knowledge inference.

There are different ontology languages available today, with a higher or lower lever of the adoption from the ontology development community, with different levels of technology maturity and toolset support. The most recent and relevant applications of ontologies are in the field of the so-called Semantic Web [16].

5.1.1 Semantic Web Ontology Languages

Semantic Web technologies define and link data on the Web (or within an enterprise system) by providing the languages to express rich, self-describing interrelations of data in a machine-processable form. Thus, machines are not only able to process long strings of characters and index tons of data, but with the Semantic Web technologies they become able to store, manage and retrieve information based on the data meaning and logical relationships (semantics). So, the semantics adds another layer to the (Web) data and is able to show, interpret and infer related facts instead of just matching the string words.

Semantic Web is based on a set of universal industrial and technology standards, as defined by the World Wide Web Consortium (W3C). The main standards that Semantic Web technology builds on are the following: XML, RDF/RDFS, OWL, and SPARQL.

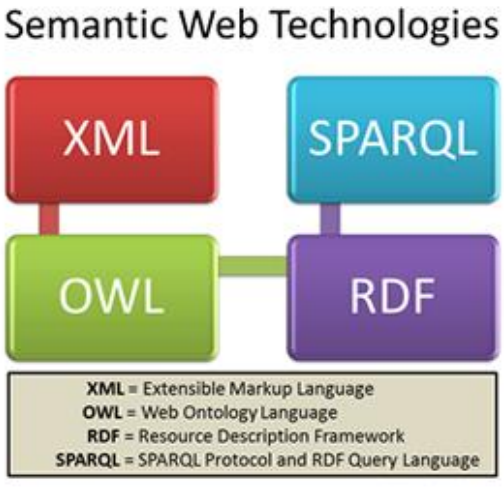


Figure 11 Semantic WEB technologies

5.1.1.1 RDF and RDFS

Resource Description Framework (RDF) is the basic language layer for data representation. RDF is the format and a data model that a semantic technology uses to describe the resources and to store the resource descriptions on the Semantic Web or in a semantic graph database. RDF describes the resources in the format of triples: subject-predicate-object.

Just as an example to illustrate the RDF language and the subject-predicate-object triple model, the listing below provides a description (and triple data model) of an imaginary resource. In the example, a RDF subject is a resource identified with '`http://www.manusquare-project.eu/data/machine/1`', while predicates are '`name`' and '`owner`', with links to '`CNC Machine`' and '`Rapid Manufacturing Ltd`' objects. The objects in this example are the plain/literal values, but the objects can be other RDF resources, which can have their own triples, again with the RDF resources as objects, and so on. Therefore, the RDF data models are in fact graphs, where RDF subjects and objects are graph nodes, and predicates are graph edges.

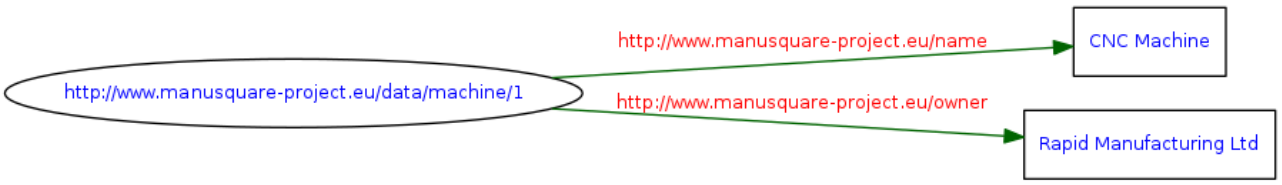
An example RDF document:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:msq="http://www.manusquare-project.eu/">

<rdf:Description rdf:about="http://www.manusquare-project.eu/data/machine/1">
  <msq:name>CNC Machine</msq:name >
  <msq:owner>Rapid Manufacturing Ltd</msq:owner >
</rdf:Description>

</rdf:RDF>
```

Triples of the RDF document:

Number	Subject	Predicate	Object
1	http://www.manusquare-project.eu/data/machine/1	http://www.manusquare-project.eu/name	"CNC Machine"
2	http://www.manusquare-project.eu/data/machine/1	http://www.manusquare-project.eu/owner	"Rapid Manufacturing Ltd"

RDF document as a graph:



Resource Description Framework Schema (RDFS) is an extension of RDF. In addition to the RDF language constructs, RDFS provides additional language constructs to define the classes and properties of the resources. In particular, RDFS provides the following constructs:

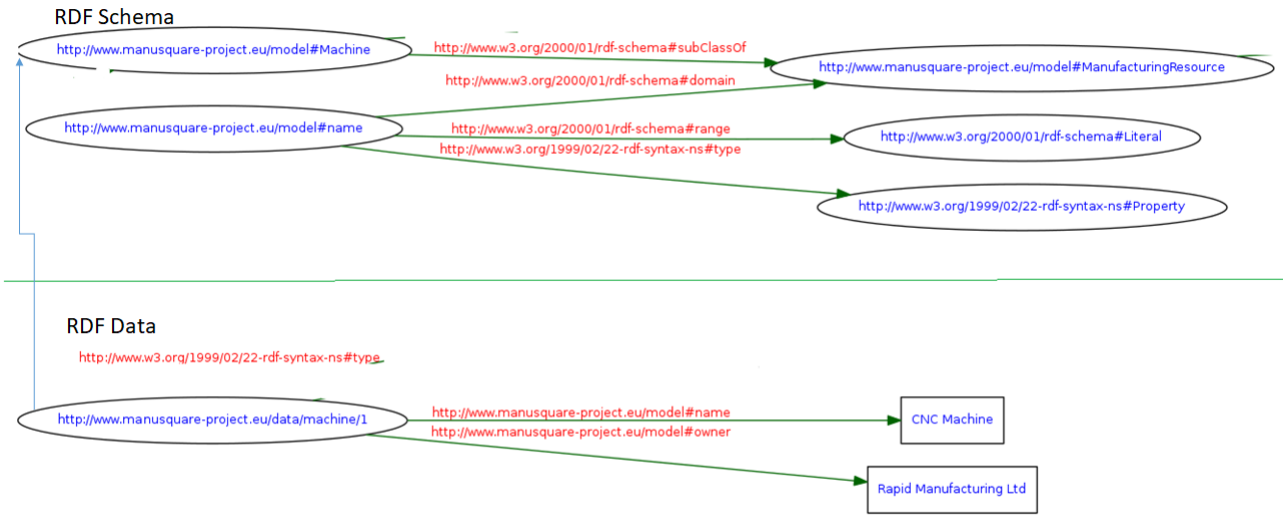
- rdfs:Class to declare/specify a class of the resources.
- rdf:Property is the class to declare the properties.
- rdfs:domain to declare the class of the subject in a triple whose predicate is a rdf:Property.
- rdfs:range to declare the class or datatype of the object in a triple whose predicate is a rdf:Property.
- rdf:type property to describe that a resource is an instance of a class.
- rdfs:subClassOf allows declaration of hierarchies of classes.

These core RDFS constructs allow for describing and embedding the semantics of user-defined vocabularies in RDF itself. Hence, by using RDFS, it is possible to structure RDF resources, by describing the classes (types) of the resources, properties of the resources, and by modelling the hierarchical subsumption of the classes and properties.

Considering our previous example of RDF description, there can be a class 'Machine' introduced into the description, to be able to link a resource '`http://www.manusquare-project.eu/data/machine/1`' to its type (to classify it), and to precisely specify a domain and range of the relevant properties. With RDFS it easy to define hierarchies of classes, e.g. that 'Machine' is a subclass of 'ManufacturingResource' class.

A RDF Schema Example (in RDF/XML syntax and as a graph):

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:msq="http://www.manusquare-project.eu/model#">

<rdfs:Class rdf:about="http://www.manusquare-project.eu/model#ManufacturingResource"/>

<rdfs:Class rdf:about="http://www.manusquare-project.eu/model#Machine">
  <rdfs:subClassOf rdf:resource="http://www.manusquare-
project.eu/model#ManufacturingResource"/>
</rdfs:Class>

<rdf:Property rdf:about="http://www.manusquare-project.eu/model#name">
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
  <rdfs:domain rdf:resource="http://www.manusquare-
project.eu/model#ManufacturingResource"/>
</rdf:Property>

<rdf:Description rdf:about="http://www.manusquare-project.eu/data/machine/1">
  <rdf:type rdf:resource="http://www.manusquare-project.eu/model#Machine"/>
  <msq:name>CNC Machine</msq:name >
  <msq:owner>Rapid Manufacturing Ltd</msq:owner >
</rdf:Description>

</rdf:RDF>
```



Therefore, the RDFS is the first, simpler Semantic Web language that can be adopted to capture the MANU-SQUARE UML conceptual model into a formal vocabulary or ontology. UML classes correspond to RDFS classes, while UML class attributes and relationships correspond to RDF properties.

5.1.1.2 OWL

The Web Ontology Language (OWL) extends RDFS and allows for expressing further schema definitions in RDF. OWL is the computational logic-based language that is designed to capture the conceptual models and that may capture also

a rich and complex knowledge about hierarchies of things (concepts, properties and individuals) and the relations between them. OWL builds on RDF and RDFS, and like them, it is constructed from XML tags, although it is worth to mention that there are other RDF syntax choices, such as: N3, turtle, etc. It is complementary to RDF and allows for formalizing a data schema/ontology in a given domain, separately from the data. Most importantly, OWL overcomes some RDFS limitations thanks to its ability to express rich semantic constructs. For example, RDFS does not allow expressions of property restrictions (value constraints and cardinality constraints) and it has very few constructs to make extensive inferences.

Similarly to RDFS, OWL allows description of classes, but using <owl:class> construct, which is actually a subclass of rdfs:Class. Unlike RDFS, OWL allows to express equality of individuals (owl:sameAs), equivalence or disjointness of properties and classes (owl:equivalentClass, owl:equivalentProperty, owl:disjointWith, owl:propertyDisjointWith), transitive properties, inverse properties, functional properties, etc. Further, OWL distinguishes between two main categories of properties: *object properties* that link individuals to individuals and *datatype properties* that link individuals to data values.

There are three versions of OWL: OWL Lite, OWL DL, and OWL Full. The main differences are related to the different logics that can be expressed with the language. More complex logics (OWL Full) enable more comprehensive descriptions and automatic inferences, but introduce limitations from the computational point of view. Therefore, the actual OWL version to be used depends on the specific application to be implemented. For example, if performances and scalability are the key aspects, and no DL reasoning is needed, then, most likely, OWL DL and OWL Full are not the best option.

5.1.1.3 SPARQL

SPARQL is the semantic query language specifically designed to query RDF/RDFS data across various systems and databases, and to retrieve and process data stored in RDF format. Therefore, it is not an ontology language per se and it will be further discussed in future deliverables, together with other languages (no standard yet) to express rules, such as Semantic Web Rule Language (SWRL).

5.1.2 Other Ontology Languages

Besides RDFS and OWL languages, there are other ontology languages. Some of them, such as DAML, OIL, DAML+OIL are predecessors of the OWL and were developed for the Semantic Web purpose.

- DAML is an Agent Markup Language based on RDF and developed by DARPA (US Defense Advanced Research Projects Agency). The DAML program ended in early 2006.
- OIL is an Ontology Inference Layer or Ontology Interchange Language, it is based on concepts developed in Description Logic (DL) and frame-languages and is compatible with RDFS. A frame language is also a technology to capture the domain knowledge or ontologies and the main constructs in the frame languages are the frames and slots. Frames can be understood as concepts, while slots can be understood as properties of the concepts.
- DAML+OIL is a successor language to DAML and OIL and has combined features of both languages. It was layered in RDF and XML. DAML+OIL was superseded by Web Ontology Language (OWL).

Other ontology languages, not strictly related to the Semantic Web initiative, include CycL, FLogic, LOOM, KIF, Ontolingua. They have been built to address specific purposes (e.g. CycL to create a rich foundational ontology or KIF to overarcing multiple, heterogeneous datasets), but no one of them is a standard and their diffusion is limited compared Semantic Web technologies.

5.2 Choices of Ontology Representation Language in other similar efforts

This section reports a summary of an analysis of previous initiatives in a manufacturing/industrial domains ontology development. The purpose of this analysis was to understand the state-of-the-art approaches for the implementation of ontologies in the manufacturing sector.

The list of initiatives/ontologies reviewed is not exhaustive. The reviewed ones are, at the best of our knowledge, the most relevant for the MANU-SQUARE project, considering the scope of the information requirements in MANU-SQUARE project. The key information concepts is MANU-SQUARE are: 1) manufacturing capability and capacity, 2) manufacturing innovation ideas, 3) manufacturing sustainability assessment, 4) request for quotation or ordering, and 5) stakeholder reputation. We will continue to seek initiatives/ontologies in the future months and in case the list will be updated. One initiative worth keeping an eye on it is the Industrial Ontology Foundry[3] Working Group.

We have not included in our analysis the foundation ontologies such as DOLCE, SUMO, or OpenCyc, although the foundational ontologies provide formally axiomatised domain independent (upper) set of concepts e.g. AbstractEntity, ConcreteEntity, Endurant, Perdurant, and spatial and temporal concepts. The issue with foundational ontologies is that they have been developed to cover everything; they are too broadly based and too generic for specific requirements of the MANU-SQUARE and similar domain-specific platforms. The foundational concepts such as AbstractEntity, ConcreteEntity, Endurant or Perdurant are indeed too generic to be used as or with the core concepts of MANU-SQUARE Ecosystem data. Their semantics of the key concepts in MANU-SQUARE are more specific and concrete than semantics of the foundational concepts, and the semantics should be unambiguous for the applications and end-users of the core concepts. Moreover, according to [18] using a foundation concept as a semantic base for manufacturing domains will group together concepts that are vastly different at the application-specific levels. Such unwanted similarities or inability to distinguish semantically different concepts may create problems for interoperability across specific manufacturing subdomains. Therefore, an ontology that is more specific than foundation ontologies and more generic than domain ontologies is required as a reference for multiple manufacturing domains [18].

Our summary is structured in Table 27. For each ontology/initiative that was reviewed, we have tried to identify an ontology type, a language used for the implementation, the focus or application scope of the ontology, and the key concepts of the ontology. Regarding the type, we distinguish between the *core manufacturing ontologies*, *domain-specific manufacturing ontologies*, and *vocabularies*. A *core ontology* provides a set of generic concepts whose semantics are shared across multiple, but not all, manufacturing domains [18]. A *domain-specific manufacturing ontology* provides a set of real-world concepts whose semantics is shared across single domain e.g. product life-cycle or process/discrete manufacturing facture. *Vocabularies* are either core or domain ontologies but without axiomatization of the ontology concepts and without semantic entailment relationships between concepts.

Ontology Name/Initiative + link	**Type (vocabulary/ domain/core)**	**Ontology Language**	**Focus/Scope**	**Notes (key concepts, real-world use of the ontology, etc.)**
Ontology Development and Utilization in Product Design [19]	Domain ontology	OWL + SWRL	Design for Manufacturing (DFM). Data Integration. Process Costs Calculation.	DFM ontology can be used to capture various manufacturing and assembly concepts and to share joining information along with the assembly geometry at the same time. In the DFM ontology, there are about ten main classes to represent the knowledge in assembly including Product, Part_material, Feature, Spatial Relationship, Degree of Freedom and Manufacturing Manufacturing Process. Most of the classes are further specialized e.g. Product is specialized into Part, Assembly, Subassembly and Platform.

[3] https://sites.google.com/view/industrialontologies/home

Manufacturing ontology for Functionally Graded Materials [20]	Domain ontology + Taxonomies (Material Taxonomy, Quality Taxonomy, Processes Taxonomy, Application Taxonomy)	OWL	To provide knowledge about the components of 'Functionally Graded Materials' (FGM) and about the manufacturing processes involved in creation of FGM.	Developed using Basic Formal Ontology (BFO) and parts of the Ontology for Biomedical Investigation (OBI). Concepts include Manufacturing Process, Material Entity, FGM Component Multiplicity, Assembly Process, etc.
MSDL-Manufacturing Service Description Language MSDL [21]	Core ontology as vocabulary	OWL DL	MSDL provides the primitive building blocks required for description of a wide spectrum of manufacturing services.	Some of the core classes of MSDL are Supplier, Customer, MfgService, Process, Industry, MfgCapability, MfgResource
FLEXINET Manufacturing Reference Ontology [22]	Core + Domain ontologies	Knowledge Frame Language (KFL)	Targeted at strategic and tactical level business decisions related to new product development and global production network configuration.	The ontology is organized into levels, starting from Level 0, which is upper ontology, going down to more specialized domain and application levels. Captures Product-Service-Production and related concepts.
MCCO - Manufacturing Core Concepts Ontology, known also as Manufacturing Reference Ontology [18], [23]	Core ontology	Knowledge Frame Language (KFL)	A manufacturing core or reference ontology (MRO) that can support the development of semantically sound application-specific ontologies for product design and production domains.	Manufacturing concepts should be categorised into eight main concept areas. These being, Realised Part, Part Version, Manufacturing Facility, Manufacturing Resource, Manufacturing Method, Manufacturing Process, Feature and Part Family. The eight categories are not claimed to be complete but they provide a step towards a complete MRO.
MASON-Manufacturing's Semantics Ontology [24]	Core ontology	OWL-DL	Draft a common semantic network in manufacturing domain.	Built upon three core concepts: Entities, Operations and Resources. - Entities provide concepts to specify the product. They give an abstract view on the product. - Operations relate to process description. They cover all processes linked to manufacturing in a wide acceptation. - Resources stand for the whole set of manufacturing linked resource, like: machine-tools, human resources, tools, etc.
A manufacturing systems interoperability ontology [25]	Core Ontology	KFL	The ontology is a foundation for a multi-level heavyweight ontology. The multiple level ontology allows increasing specificity (to ensure accuracy), while the heavyweight (i.e. computer interpretable) nature provides the semantic and logical, rigour required.	The ontology includes terms that are structured around potential core concepts such as: Metric, Data, Constraint, Target, Response, Timescale, Manufacturing Method, Interface, Visualisation, Collaboration, Prediction, Authority Level, Standard Sustainment, Person, Analysis, Traceability, Item, Status, System, Resource. The ontology has been developed for a PhD Thesis and it seems that it did not have concrete applications

				(beyond the ontology experimentations).
The QUDT - Quantities, Units, Dimensions and Data Types Ontologies (QUDT) [26]	Core, cross-domain Ontology	Multiple formalisations, including OWL	Domain independent. Unified model of, measurable quantities, units for measuring different kinds of quantities, the numerical values of quantities in different units of measure and the data structures and data types used to store and manipulate these objects in software.	Defines the core classes and properties, and restrictions used for modelling physical quantities, units of measure, and their dimensions in various measurement systems.
e-Kanban Ontology [27]	Domain Ontology	OWL	It describes key concepts needed to support vendor managed inventory as identified by the Auto Industry Action Group (AIAG) and supporting the Open Applications Group Integration Specification (OAGIS).	Main elements of the ontology include descriptions of datatypes and classes about: Processes and Roles, Identifiers, Parties, Documents, Locations, Items, Shipment and Shipment schemas, Carriers and Equipment, Kanbans, Messages. This ontology was successfully used in a FP7 ATHENA project to automate interoperability of different inventory management systems.
CEN/WS SERES Ontology [28]	Domain Ontology	RDFS/OWL	Ontology that can be used to exchange data on the description of engineering materials.	The ontology captures concepts about materials and documents (that are used to describe such materials). For example, the ontology enables to describe engineering material types, engineering material characterizations, engineering material processes, engineering material sources, customer orders and works orders.
SatisFactory Semantic Layer [29]	Domain Ontology	RDFS	Ontology to support automatic analysis and design of dynamically evolving shop floor operations.	Ontology-based context model which captures the general concepts about user and business context Some knowledge is statically stored (knowledge coming from domain experts) and some knowledge is derived in real time from data coming from shop floor sensors.
Enterprise Ontology [30]	Domain Ontology	Ontolingua	A collection of terms and definitions relevant to business enterprises.	-
AutomationML Ontology [31]	Vocabulary	OWL	Automation, plant description.	AutomationML ontology (AMLO) covers the CAEX part of the AutomationML standard. It allows industry to interlink and integrate heterogeneous data more efficiently and to benefit from the Semantic Web tools and technology stack.

Table 27 Review of ontology representation languages in similar efforts

As reported in , there are many different initiatives for manufacturing sector ontology development in the literature. They differ from each other in the type of ontology that has been implemented (core vs. domain vs. vocabulary / etc.), in the

ontology language that has been used to formalise the ontology, and in the scope or coverage of concepts. The analysis was important for two reasons:

- it reveals state of the practice in encoding/representing the manufacturing ontologies;
- it reveals what are the core manufacturing concepts in other works.

As can be seen from the table, most of the ontologies are encoded using a Semantic Web technologies. Especially, that is the case with ontologies that were used in a 'real-world' demonstrations and implementations (e.g. e-Kanban Ontology in FP7 ATHENA project). Most of the key/core concepts of those reviewed ontologies are also the key concepts in the MANU-SQUARE, which opens up the possibilities for integration and alignment of MANU-SQUARE ecosystem with existing ontology initiatives in the manufacturing space.

5.3 Ontology Representation Language Requirements for MANU-SQUARE Core Ontology

Before making a final step, which is the formalisation of the MANU-SQUARE core conceptual models into an ontology using an ontology language, a precise set of requirements that should drive the choice about the ontology language and desired richness of the semantics in the MANU-SQUARE core ontology is defined. Requirements are described as follows.

Req 1. **Standard language and interoperability**. MANU-SQUARE will be dealing with an industry and one key element for industries to adopt a new technology, such is MANU-SQUARE, is the use of standards. If the MANU-SQUARE ontology will follow a standard language representation, then it could be better accepted by the industry.

Req 2. **Reusability.** MANU-SQUARE does not want to reinvent the wheel. As introduced above, other manufacturing and industrial ontologies are available, with manufacturing domain concepts (e.g. process, resource, items) and domain-independent concepts (e.g. quantities and unit of measures) applicable to MANU-SQUARE. Therefore, whenever possible and when system requirements clearly state the need, the project should aim to reuse existing ontologies, trying to reduce as much as possible the (most likely necessary) adaptations/customisations. To make the reuse existing ontologies feasible and to align MANU-SQUARE ontologies with them, the MANU-SQUARE ontology has to be based on the same ontology language as other ontologies.

Req 3. **Shareability and Extensibility**. MANU-SQUARE is a R&D project with precise budget and resource allocation, thus, the creation of an ontology that covers all possible domains/services is out of scope. The project has to develop a core set of conceptual models and a set of methodologies and tools to progressively extend and adapt such a core representation to multiple settings. The project will demonstrate the core representation in a few selected use-cases; but then third parties should be able to start from the project outputs and derive new representations. Therefore, the selected ontology language, as well as the associated tools and development methodologies, should be able to consistently describe and specify any element of a current or future system or the systems in its entirety including its concepts, relationships, processes, interactions and interoperation and must work across the different organisation levels. This must include new concepts or types of relationship.

Req 4. **Open source and costs**. Linked to the previous point, but more focused on the tools (software) that will manage the MANU-SQUARE ontology, the project aims to deal with the open source and free of charge licences. This approach has some benefits:

- the project aims to use existing open source solutions to develop MANU-SQUARE modules and services; in this way, the MANU-SQUARE platform can benefit from existing communities around open source projects.
- the project aim to deliver open source tools to facilitate the future exploitation of developed tools. This approach is also linked to the idea of creating a specific community around the envisioned MANU-SQUARE platform.

- In addition, it should be considered that one of the key stakeholder groups of MANU-SQUARE are SMEs, which have limited resources and thus cannot invest a lot on (not core) new technologies. The use of open source can indeed address the cost barriers for them.

Req 5. **Scalability.** MANU-SQUARE aims to become a B2B cloud platform involving many customers across Europe. In this context, the scalability aspect is a key element to be taken into consideration. It affects all architectural elements of the platform and, thus, also the semantic layers and ontology language/framework decision. There is also a trade-off between highly expressive languages and computational speed. The actual need for complex inferences and reasoning (and thus highly expressive languages) in the project scenarios and platform services should be carefully taken into consideration.

5.4 MANU- SQUARE Core Ontology Implementation

Finally, the UML MANU-SQUARE conceptual models introduced in the previous sections are translated into their formal and machine-interpretable ontological representation. As a representation language for the MANU-SQUARE Core ontology the RDFS and OWL Lite language from the Semantic Web technology stack are chosen.

The reason for using the RDFS/OWL languages is somehow obvious, but it is strongly based on our analysis of existing ontologies (§ 5.2) and MANU-SQUARE specific requirements (§ 5.3). Clearly, RDFS and OWL are standard languages, supported by the W3C community. There are open source, free of charge, and mature tools for RDF, RDFS and OWL ontologies and data management. Existing industrial ontologies are based on RDFS and OWL too. Even the scalability may be adequately addressed in the future, as there are commercial tools that, compared to some free of charge tools, claim to provide better performance and scalability for the reasoning and querying over the RDF-based knowledge graph database.

Hence, the ontology is implemented in both RDFS and OWL languages and published in a RDF/XML and Turtle syntax (please see Appendix A). This idea of creating the ontology by combining RDFS and OWL is adopted from a well-known http://xmlns.com/foaf/spec/ ontology[4]. FOAF is captured in both RDFS and OWL. It seems to us that the approach taken by the FOAF is very pragmatic also for the MANU-SQUARE project, as the project use-cases and inference/reasoning requirements are not currently developed with enough detail to make a final decision about the RDFS versus OWL Lite versus OWL DL expressivity.

Therefore, the MANU-SQUARE platform now provides a RDFS version of the MANU-SQUARE Core Ontology, and such representation is for sure a starting point of the implementation and for the development of Task 2.2, Task 2.3 and Task 2.4, but also provides an OWL version of the MANU-SQUARE Core Ontology, in a case that OWL/DL reasoning will be required in the future. This “dual” ontology representation gives a nice flexibility to proceed with the project and technology implementation, and if needed in a later stage of the project, to gradually increase semantics expressiveness of the MANU-SQUARE Core and domain/tool-specific ontologies.

We call this ontology MANU-SQUARE Core Ontology, as it provides a set of generic concepts that are shared across multiple manufacturing domains and applications dealing with the concepts of (1) manufacturing capability and capacity, (2) manufacturing innovation ideas, (3) manufacturing sustainability assessment, (4) request for quotation or ordering, and (5) ecosystem stakeholders’ reputation.

The reason why this ontology can be considered a core ontology is found in the literature. According to [31] the core ontologies in manufacturing sector cover concepts such as process, product, and resource. In addition to providing formal specifications of the semantics of these generic concepts, core ontologies are also designed to maximize shareability and reusability, and hence do not make any ontological commitments that are not shared by all related domain ontologies. That is exactly the scope and purpose of MANU-SQUARE Core Ontology. It describes the core

[4] FOAF – Friend of a Friend Ontology http://xmlns.com/foaf/spec/

concepts and their relations of the MANU-SQUARE platform; it will be the conceptual "glue" of the several area-specific models (ontologies).

The actual translation of UML representation of MANU-SQUARE conceptual models into the formalized RDFS/OWL ontology was done by applying the translation rules detailed in Table 28.

UML Class	<rdfs:class>, <owl:class>
UML Class inheritance	<rdfs:subClassOf>
UML Primitive Data Type Attribute	<rdf:property> and < owl:DatatypeProperty> with a proper <rdfs:domain> and <rdfs:range>. For example, if the UML attribute data type is String, the property range is a plain literal <rdf:Literal>, or if the UML attribute data type is decimal, the property range is a typed literal <xsd:decimal>
UML Relationship	<rdf:property> and <owl:ObjectProperty> with a proper <rdfs:domain> and <rdfs:range> expression. E.g. <rdfs:class> of an UML Class that owns the relationship is a property domain (<rdfs:domain>) and another UML class that is at the end of relationship is the propery range (<rdfs:range>)
UML Instance	rdf resourse with appropriate <rdf:type>

Table 28 UML-to-Ontology transformation rules

An automated UML-to-Ontology transformation tool is implemented in order to make the MANU-SQUARE UML transformation to Ontology more accurate (less error-prone) and to make sure that the transformation process can be easily repeatable in a case UML models change. The transformation is driven by the transformation rules in Table 28.

As illustrated in Figure 12, the tool takes an XMI as an input and provides as an output RDFS/OWL representation of the MANU-SQUARE Core Ontology. XMI stands for an XML Metadata Interchange (XMI)[5] and it is a standard interchange format for UML models. The transformation tool is supported by the UML2 Java library of Eclipse Foundation Modelling Framework[6] that implements the meta-model of the Unified Modelling Language (UMLTM) 2.x OMG.

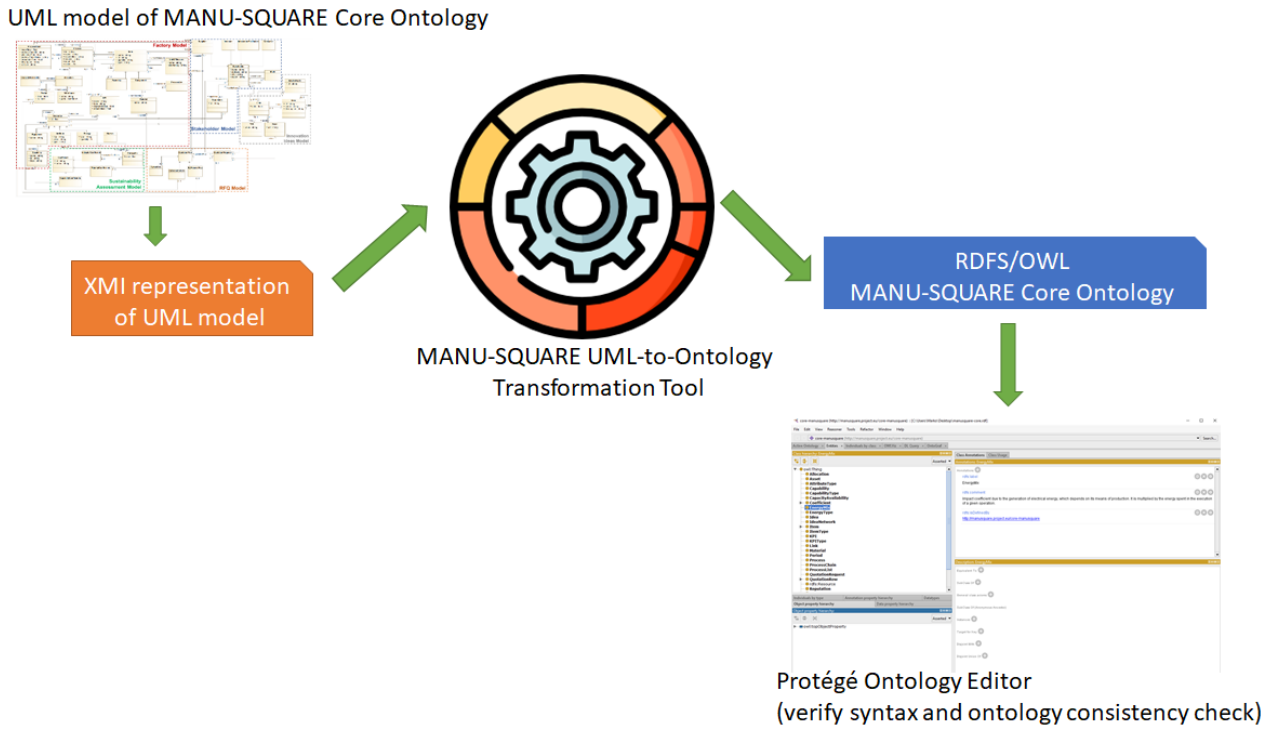


Figure 12 MANU-SQUARE UML to Ontology Transformation process

[5] Open Management Group XMI standard - https://www.omg.org/spec/XMI/About-XMI/
[6] EMF UML2 - http://www.eclipse.org/modeling/mdt/?project=uml2

For example, the 'Process' UML class and 'input', 'output' UML relationships are translated into `owl:class Process`, and two object properties `hasInput`, `hasOutput`.

```
<owl:Class rdf:about="http://manusquare.project.eu/core-manusquare#Process">
    <rdfs:comment>Process represents the more significant class in the factory
description. It aggregates the management materials/sub-products/products parts with the
resources necessary to produce them. An example of process could be the milling process
that transform a work piece removing part of its material. </rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="http://manusquare.project.eu/core-manusquare"/>
    <rdfs:label>Process</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 </owl:Class>


<owl:ObjectProperty rdf:about="http://manusquare.project.eu/core-manusquare#hasOutput">
    <rdfs:isDefinedBy rdf:resource="http://manusquare.project.eu/core-manusquare"/>
    <rdfs:range rdf:resource="http://manusquare.project.eu/core-manusquare#Item"/>
    <rdfs:domain rdf:resource="http://manusquare.project.eu/core-manusquare#Process"/>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://manusquare.project.eu/core-manusquare#hasInput">
    <rdfs:isDefinedBy rdf:resource="http://manusquare.project.eu/core-manusquare"/>
    <rdfs:range rdf:resource="http://manusquare.project.eu/core-manusquare#Item"/>
    <rdfs:domain rdf:resource="http://manusquare.project.eu/core-manusquare#Process"/>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </owl:ObjectProperty>
```

The output of the tool is a RDFS/OWL MANU-SQUARE Core Ontology. The ontology was further opened in an ontology editor tool Protégé[7] to check correctness of the syntax and ontology. A consistency of the ontology was checked / verified with a Protégé built-in reasoner.

For the readers' convenience, Table 29 reports a first list of core concepts and properties for the MANU-SQUARE Core Ontology.

MANU-SQUARE Core Concepts	MANU-SQUARE Core Properties
Allocation	hasActualValue
Assembly	hasAdditionalTime
Asset	hasAddress
Attribute	hasAllocation
AttributeType	hasAsset
ByItemRow	hasAttribute
ByProcessRow	hasCalendar
ByResourceRow	hasCapability
Capability	hasCity
CapabilityCoefficient	hasCost
CapabilityType	hasDate
CapacityAvailability	hasDescription
Coefficient	hasEnergy
Component	hasEnergyMix
Consumer	hasExpectedValue

[7] https://protege.stanford.edu/

Energy	hasFrom
EnergyMix	hasId
EnergyType	hasIdea
Equipment	hasIndicatorCoefficent
Human	hasInput
Idea	hasInvestedAmount
IdeaNetwork	hasItem
IndicatorCoefficient	hasKey
InnovationFacilitator	hasKpi
Investor	hasLink
Item	hasMaterial
ItemType	hasMeanTime
KPI	hasName
KPIType	hasNation
Link	hasOperation
Material	hasOutput
MaterialCoefficient	hasParts
Multi	hasPeriod
Period	hasProcess
Process	hasProcessList
ProcessChain	hasQuantity
ProcessList	hasQuotationRow
QuotationRequest	hasReputation
QuotationRow	hasRequestee
Reputation	hasRequester
Resource	hasResource
Stakeholder	hasShortening
Supplier	hasStakeholder
TAG	hasSupplier
UnitOfMeasure	hasTag
Warehouse	hasThroughputTime hasTo, hasUnitOfMeasure,,hasUrl,,hasValue,hasVersion hasWarehouse

Table 29 List of MANU-SQUARE core concepts and properties

Finally, Figure 13, Figure 14 and Figure 15 show several screenshots of the MANU-SQUARE Core ontology as opened in Protégé Ontology Editor.

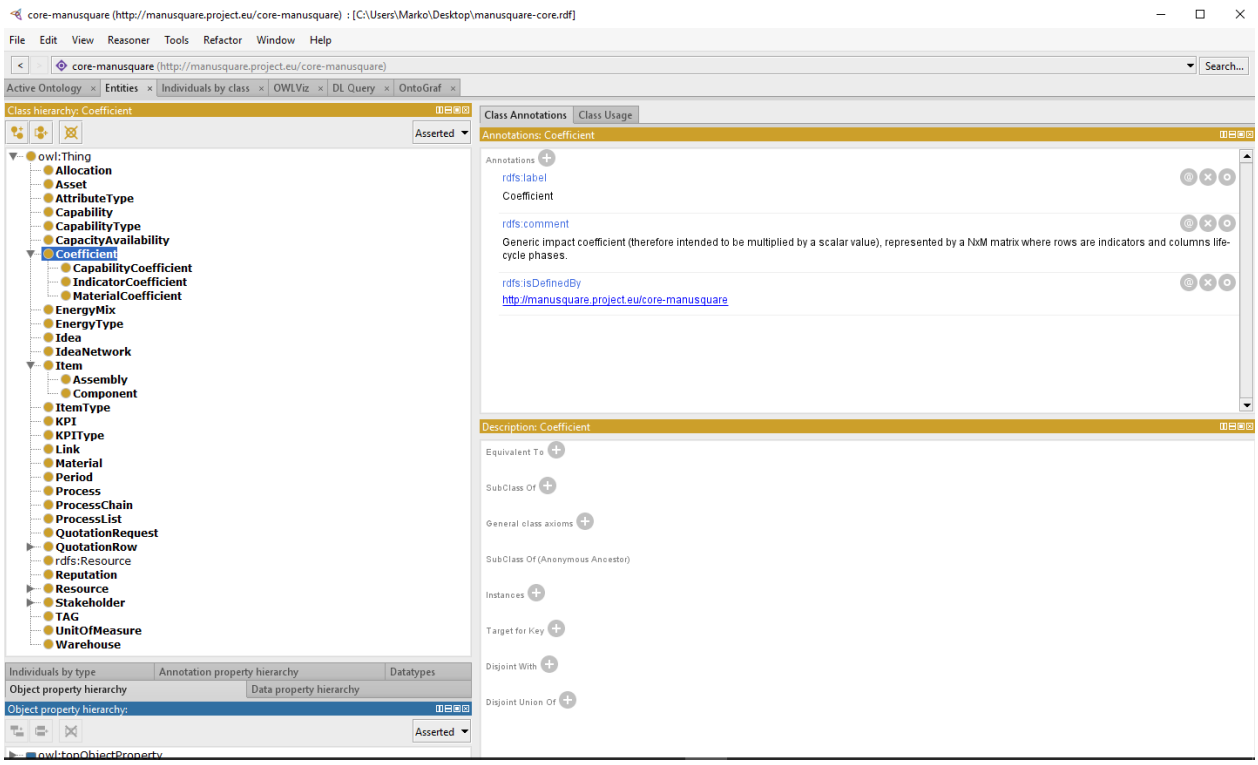


Figure 13 MANU-SQUARE Core Ontology shown in Protégé: Class hierarchy

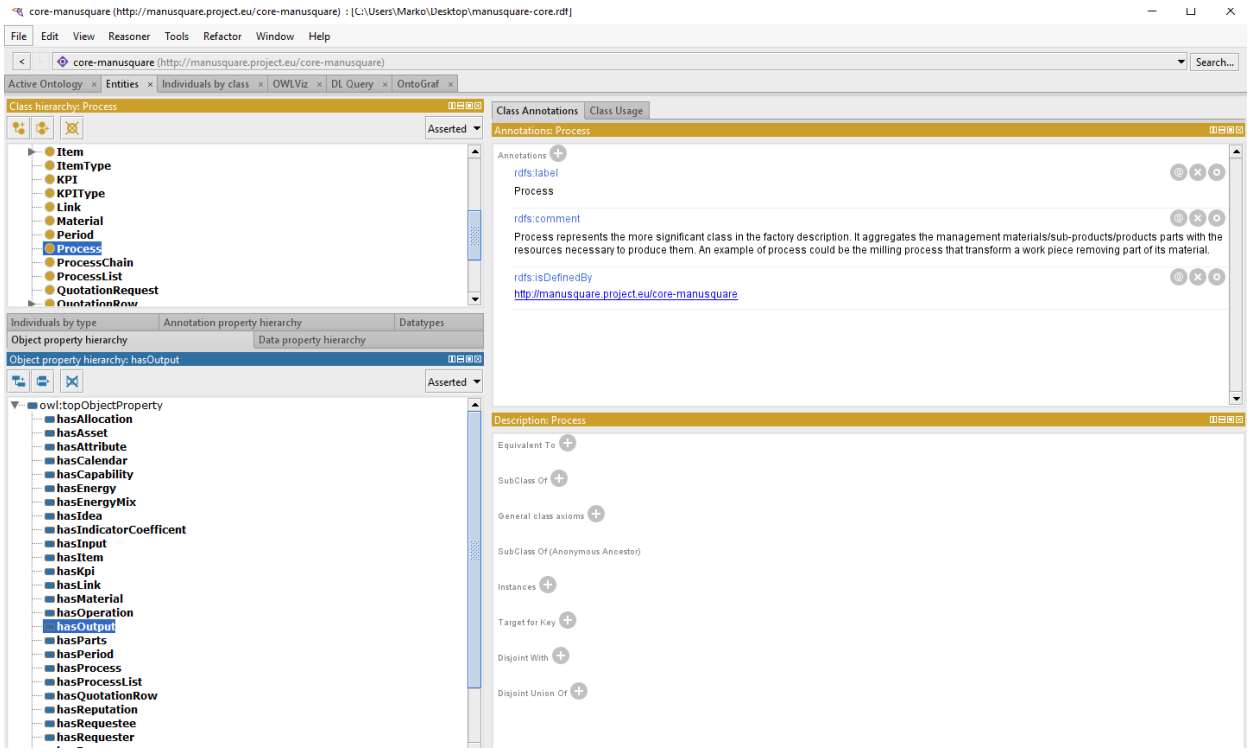


Figure 14 MANU-SQUARE Core Ontology shown in Protégé: Object Properties

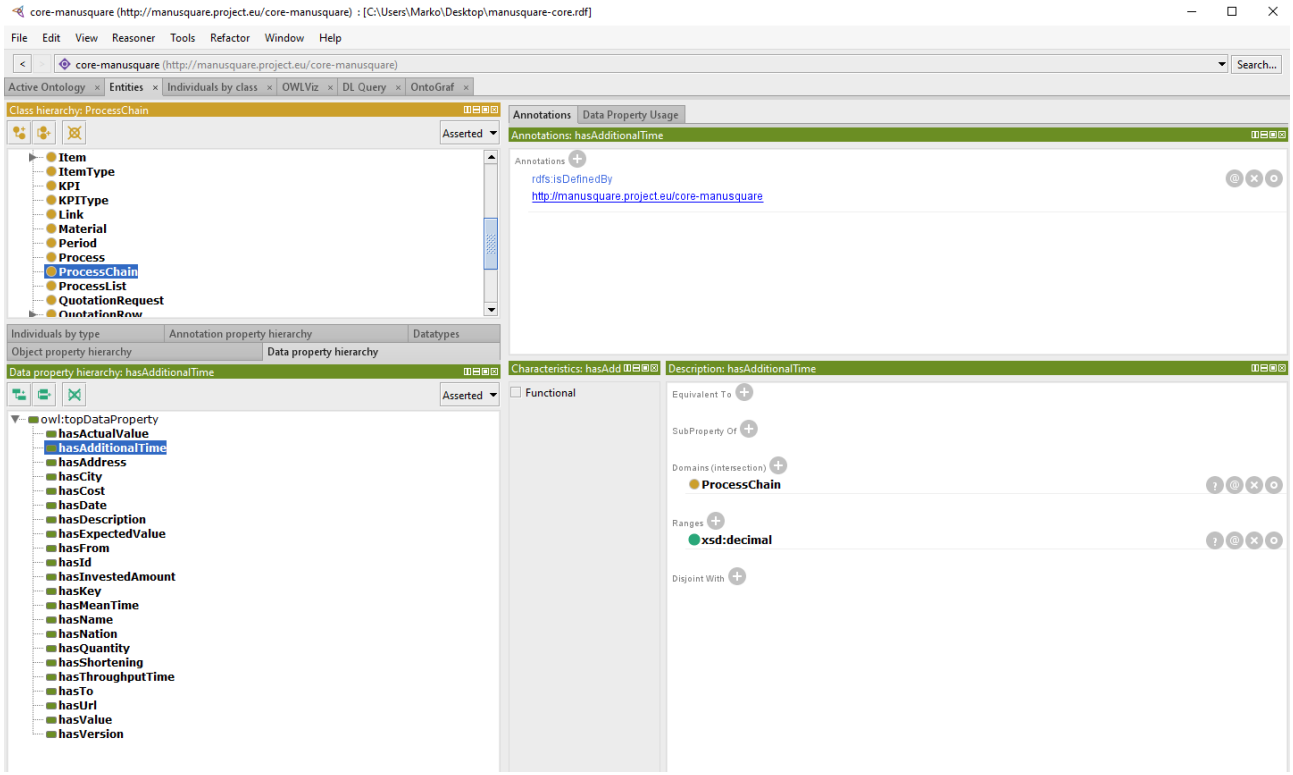


Figure 15 MANU-SQUARE Core Ontology shown in Protégé: Datatype Properties

As a final word about the MANU-SQUARE Core Ontology implementation, it should be mentioned that the ontology is expected to further evolve and to be changed to some extent in the near future, as there will be new requirements coming from the project tasks.

6 CONCLUSION

This deliverable presents a technical description of the Ecosystem Data Model, considering the aspects related to core abstract concepts of the MANU-SQUARE platform.

The data model formalization has been based on a well-defined approach, here reported, which started from a standard representation of the Ecosystem Data Model, by using UML technique, to the development of its ontology representation which represents the main goal of the Task 2.1.

The data model described in this document poses the basis for the next activities where the data model will be extended and contextualized in each industrial sector whilst it will support the functioning of the service-providing applications belonging to the MANU-SQUARE platform.

Since a continuous improvement of the Ecosystem Data Model is expected, the present version of the document represents the starting base of future releases of the application that will be kept up to date and circulated to the involved MANU-SQUARE members in order to enable an effective usage of the model.

REFERENCE LIST

[1] Chen, P.: The entity-relationship model—toward a unified view of data, In: Proceeding on the International Conference on very Large Data Bases: 22-24 September, 1975, Framingham, p. 9 – 36, 1976

[2] Chen, P.: Entity-Relationship Approach to Information Modeling and Analysis. In: Proceedings of the Second International Conference on the Entity-Relationship Approach (ER'81), Washington, DC, USA, 12-14 October, 1981. North-Holland 1983

[3] Thalheim, B.: Entity-relationship modeling: Foundations of database technology, 2000

[4] Beynon-Davies, P.: Database Systems. Houndmills, Basingstoke, UK: Palgrave, 2004

[5] Bézivin, J.; Muller, P.-A.: The unified modeling language: UML '98: beyond the notation, 1999

[6] McCaleb, M. R.: A Conceptual Data Model of Datum Systems; National Institute of Standards and Technology, 1999

[7] ISO 10303-11:2004: Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual, 2004

[8] STEP Tools Incorporated: EXPRESS-G Language Overview, 2008

[9] IDEF - Family of Methods – A Structured Approach to Enterprise Modeling and Analysis, available at: http://www.idef.com/, 2009-10-16

[10] Bernus, Peter: Handbook on Architectures of Information Systems, Springer, 2005

[11] Halpin, T.; Morgan, T.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design, Morgan Kaufmann, 2008

[12] Halpin, T.: Object-Role Modeling; Neumont University, USA, 2007, available at: http://www.orm.net/pdf/EncDBS.pdf, 2009-11-16.

[13] Halpin, T.: ORM 2; Paper Neumont University, Salt Lake City, Utah, USA. 2005

[14] Clark, D.: Database Modeling with Object Role Modeling. In: Net-article, SYS-CON Media, Inc., 2004, available at: http://dotnet.sys-con.com/node/45133, 2009-11-25

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

[16] Sematic Web Web page https://www.w3.org/standards/semanticweb/, accessed Aug 2018.

[17] Deshayes, Laurent, Sebti Foufou, and Michael Gruninger. "An ontology architecture for standards integration and conformance in manufacturing." Advances in Integrated Design and Manufacturing in Mechanical Engineering II. Springer, Dordrecht, 2007. 261-276.

[18] Usman, Zahid, et al. "Towards a formal manufacturing reference ontology." International Journal of Production Research 51.22 (2013): 6553-6572.

[19] Chang, Xiaomeng. Ontology Development and Utilization in Product Design. Diss. Virginia Tech, 2008.

[20] Furini, Francesco, et al. "Development of a manufacturing ontology for functionally graded materials." ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2016.

[21] 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.

[22] FLEXINET Deliverable D3.5 Product-service-production ontologies report, 2016. FP7-2013-NMP-ICT-FOF (RTD) Project 608627. Esmond Urwin (LU), Claire Palmer (LU), Bob Young (LU), Ester Palacios (ITI), Jose Miguel Pinazo (AINIA) Avelino Font (AINIA), Raquel Almarcha (AINIA)

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

[24] Severin Lemaignan, et al. “MASON: A Proposal For An Ontology Of Manufacturing Domain”, 2006 https://academia.skadge.org/publis/Lemaignan2006.pdf

[25] A manufacturing systems interoperability ontology. A formal ontology for new system requirements evaluation. Hastilow, Neil. PhD thesis. 2013. https://dspace.lboro.ac.uk/dspace-jspui/handle/2134/13174

[26] QUDT - Quantities, Units, Dimensions and Data Types Ontologies, http://www.qudt.org/release2/qudt-catalog.html, 2017

[27] Barkmeyer, E. J. An Ontology for the e-kanban Business Process. National Institute of Standards and Technology. Internal Report 7404, 2007. https://doi.org/10.6028/NIST.IR.7404

[28] CEN/WS SERES ICT Standards in Support of an eReporting Framework for the Engineering Materials Sector

[29] SATISFACTORY Semantically-enriched framework for analysis and design of dynamically evolving shop floor operations: http://www.satisfactory-project.eu/satisfactory/wp-content/uploads/2016/10/SatisFactory-D3.1-v1.0-Semantically-enriched-framework.pdf

[30] Uschold, Mike, et al. "The enterprise ontology." The knowledge engineering review 13.1 (1998): 31-89. The Enterprise Ontology http://www.aiai.ed.ac.uk/project/enterprise/ontology.html

[31] Deshayes, Laurent, Sebti Foufou, and Michael Gruninger. "An ontology architecture for standards integration and conformance in manufacturing." Advances in Integrated Design and Manufacturing in Mechanical Engineering II. Springer, Dordrecht, 2007. 261-276.

APPENDIX A : MANU-SQUARE CORE ONTOLOGY (IN TURTLE SYNTAX)

```
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix msq:   <http://manusquare.project.eu/core-manusquare#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix so:    <http://schema.org/> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

<http://manusquare.project.eu/core-manusquare>
        a               owl:Ontology ;
        rdfs:comment      "ManuSquare Core Ontology covers five areas of the manufacturing: (1)
      Factory Model through the classes needed to provide a description of a company, manufacturing
      processes and involved resources,(2) Stakeholder Model with the concepts of Stakeholder
      (Supplier, Investor, Innovation Facilitators, Customers),(3) Innovation Ideas Model for
      capturing different innovate ideas in manufacturing domain, linking them with their
      stakeholders, (4) RFQ Model with classes required to define the concept of Request for
      Quotation, and (5) Sustainability Assessment Model with  classes needed to perform evaluation
      of the processes from a sustainability point of view." , "version 0.9.20190731" ;
        dc:contributor  "Gabriele Izzo" , "Marko Vujasinovic" , "Andrea Barni" , "Andrea Bettoni" ,
      "Giuseppe Landolfi" , "Alessio Gugliotta" ;
        dc:publisher    "MANU-SQUARE Horizon 2020 Project " ;
        dc:rights       "https://creativecommons.org/licenses/by/3.0/" ;
        dc:title        "Manusquare Core Ontology" ;
        foaf:mbox       "manusquare-project@innova-eu.net" .


msq:ItemType  a           owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ItemType" .

msq:hasExpectedValue  a   owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:KPI ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .

msq:hasAllocation  a      owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:CapacityAvailability ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Allocation .

msq:hasDate  a            owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Link ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:dateTime .

msq:Warehouse  a          owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Warehouse" .

msq:hasTo  a              owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Period ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:dateTime .

msq:hasFrom  a            owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Period ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:dateTime .

msq:Link  a               owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Link" .

msq:hasTag  a             owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:TAG .

msq:Reputation  a         owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Reputation" .
```

```
msq:hasIdea  a            owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Idea .

msq:CapacityAvailability
        a                 owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "CapacityAvailability" .

msq:Coefficient  a        owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Coefficient" .

msq:hasShortening  a      owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:UnitOfMeasure ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:CapabilityType  a     owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "CapabilityType" .

msq:hasQuantity  a        owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:Component  a          owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Component" ;
        rdfs:subClassOf   msq:Item .

msq:hasRequester  a       owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:QuotationRequest ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Stakeholder .

msq:Stakeholder  a        owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Stakeholder" .

msq:Capability  a         owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Capability" .

msq:Attribute  a          owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Attribute" ;
        rdfs:subClassOf   msq:Resource .

msq:hasKpi  a             owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:KPI .

msq:ByResourceRow  a      owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ByResourceRow" ;
        rdfs:subClassOf   msq:QuotationRow .

msq:hasIndicatorCoefficent
        a                 owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:EnergyMix ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:IndicatorCoefficient .

msq:hasCity  a            owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Stakeholder ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:hasName  a            rdf:Property , owl:DatatypeProperty ;
        rdfs:domain       owl:Thing , rdfs:Resource ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .
```

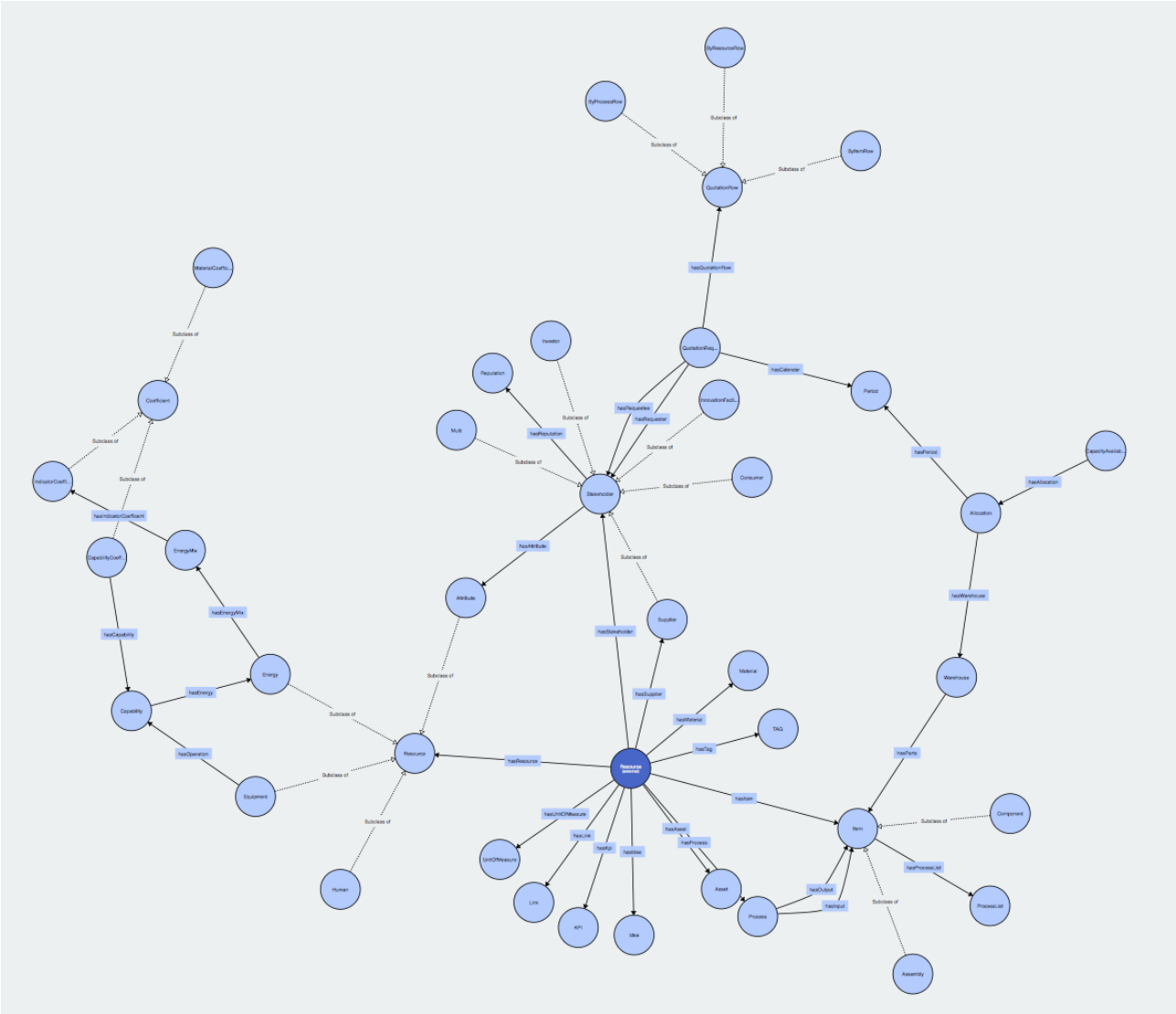
```
msq:hasMaterial  a          owl:ObjectProperty , rdf:Property ;
        rdfs:domain         rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:Material .

msq:Idea  a                 owl:Class , rdfs:Class ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label          "Idea" .

msq:hasEnergyMix  a         owl:ObjectProperty , rdf:Property ;
        rdfs:domain         msq:Energy ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:EnergyMix .

msq:hasKey  a               owl:DatatypeProperty , rdf:Property ;
        rdfs:domain         msq:Attribute ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          rdfs:Literal .

msq:hasUnitOfMeasure  a     owl:ObjectProperty , rdf:Property ;
        rdfs:domain         rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:UnitOfMeasure .

msq:hasItem  a              owl:ObjectProperty , rdf:Property ;
        rdfs:domain         rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:Item .

msq:hasCost  a              owl:DatatypeProperty , rdf:Property ;
        rdfs:domain         msq:EnergyMix ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          xsd:decimal .

msq:hasUrl  a               owl:DatatypeProperty , rdf:Property ;
        rdfs:domain         msq:Asset ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          rdfs:Literal .

msq:Investor  a             owl:Class , rdfs:Class ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label          "Investor" ;
        rdfs:subClassOf     msq:Stakeholder .

msq:hasType  a       rdf:Property ;
        rdfs:domain  rdfs:Resource , owl:Thing .

msq:Equipment  a            owl:Class , rdfs:Class ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label          "Equipment" ;
        rdfs:subClassOf     msq:Resource .

msq:hasStakeholder  a       owl:ObjectProperty , rdf:Property ;
        rdfs:domain         rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:Stakeholder .

msq:hasSupplier  a          owl:ObjectProperty , rdf:Property ;
        rdfs:domain         rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:Supplier .

msq:hasRequestee  a         owl:ObjectProperty , rdf:Property ;
        rdfs:domain         msq:QuotationRequest ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range          msq:Stakeholder .

msq:CapabilityCoefficient
        a                   owl:Class , rdfs:Class ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label          "CapabilityCoefficient" ;
        rdfs:subClassOf     msq:Coefficient .

msq:Assembly  a             owl:Class , rdfs:Class ;
        rdfs:isDefinedBy    <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label          "Assembly" ;
        rdfs:subClassOf     msq:Item .
```

```
msq:hasAsset  a           owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Asset .

msq:ProcessList  a        owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ProcessList" .

msq:Allocation  a         owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Allocation" .

msq:Consumer  a           owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Consumer" ;
        rdfs:subClassOf   msq:Stakeholder .

msq:InnovationFacilitator
        a                 owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "InnovationFacilitator" ;
        rdfs:subClassOf   msq:Stakeholder .

msq:IndicatorCoefficient
        a                 owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "IndicatorCoefficient" ;
        rdfs:subClassOf   msq:Coefficient .

msq:Human  a              owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Human" ;
        rdfs:subClassOf   msq:Resource .

msq:Process  a            owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Process" .

msq:Material  a           owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Material" .

msq:hasThroughputTime
        a                 owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:ProcessChain ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .

msq:hasParts  a           owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Warehouse ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Item .

msq:TAG  a                owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "TAG" .

msq:hasReputation  a      owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Stakeholder ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Reputation .

msq:hasQuotationRow  a    owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:QuotationRequest ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:QuotationRow .

msq:Asset  a              owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Asset" .

msq:hasResource  a        owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Resource .

msq:KPIType  a            owl:Class , rdfs:Class ;
```

```
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "KPIType" .

msq:hasNation  a          owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Stakeholder ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:Supplier  a           owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Supplier" ;
        rdfs:subClassOf   msq:Stakeholder .

msq:QuotationRow  a       owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "QuotationRow" .

msq:hasOutput  a          owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Process ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Item .

msq:hasWarehouse  a       owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Allocation ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Warehouse .

msq:hasDescription  a     owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:hasAddress  a         owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Stakeholder ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:hasActualValue  a     owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:KPI ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .

msq:ByProcessRow  a       owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ByProcessRow" ;
        rdfs:subClassOf   msq:QuotationRow .

msq:hasCapability  a      owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:CapabilityCoefficient ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Capability .

msq:Item  a               owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Item" .

msq:Multi  a              owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Multi" ;
        rdfs:subClassOf   msq:Stakeholder .

msq:EnergyMix  a          owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "EnergyMix" .

msq:Energy  a             owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Energy" ;
        rdfs:subClassOf   msq:Resource .

msq:Period  a             owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Period" .

msq:hasLink  a            owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Link .
```

```
msq:hasProcessList  a     owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Item ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:ProcessList .

msq:KPI  a                owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "KPI" .

msq:hasProcess  a         owl:ObjectProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Process .

msq:hasValue  a           owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:hasAdditionalTime
        a                 owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:ProcessChain ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .

msq:hasInput  a           owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Process ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Item .

msq:Resource  a           owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "Resource" .

msq:hasAttribute  a       owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Stakeholder ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Attribute .

msq:hasOperation  a       owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Equipment ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Capability .

msq:hasPeriod  a          owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Allocation ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Period .

msq:hasEnergy  a          owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:Capability ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Energy .

msq:hasCalendar  a        owl:ObjectProperty , rdf:Property ;
        rdfs:domain       msq:QuotationRequest ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        msq:Period .

msq:AttributeType  a      owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "AttributeType" .

msq:EnergyType  a         owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "EnergyType" .

msq:IdeaNetwork  a        owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "IdeaNetwork" .

msq:hasInvestedAmount
        a                 owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:Investor ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .
```

```
msq:QuotationRequest  a   owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "QuotationRequest" .

msq:UnitOfMeasure  a      owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "UnitOfMeasure" .

msq:ProcessChain  a       owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ProcessChain" .

msq:hasId  a              owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:hasMeanTime  a        owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       msq:ProcessChain ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        xsd:decimal .

msq:hasVersion  a         owl:DatatypeProperty , rdf:Property ;
        rdfs:domain       rdfs:Resource , owl:Thing ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:range        rdfs:Literal .

msq:ByItemRow  a          owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "ByItemRow" ;
        rdfs:subClassOf   msq:QuotationRow .

msq:MaterialCoefficient
        a                 owl:Class , rdfs:Class ;
        rdfs:isDefinedBy  <http://manusquare.project.eu/core-manusquare> ;
        rdfs:label        "MaterialCoefficient" ;
        rdfs:subClassOf   msq:Coefficient .
```


ByResourceRow
ByProcessRow
ByItemRow
QuotationRow
Subclass of
hasQuotationRow
MaterialCostRo...
CostRoleInt
Investor
Reputation
Multi
QuotationReq...
hasCalendar
Period
InnovationFacto...
hasReputation
hasRequestee
hasRequester
Consumer
Stakeholder
hasPeriod
CapacityAvailab...
hasAllocation
Allocation
IndicatorCostR...
hasIndicatorCostRoleInt
EnergyMix
CapacityCostI...
hasAttribute
Attribute
Supplier
hasWarehouse
hasEnergyMix
hasCapacityUnit
hasStakeholder
Warehouse
Energy
hasSupplier
Material
hasMaterial
TAG
hasEnergy
Capability
hasTag
hasPorts
hasOperation
Resource
hasResource
Equipment
hasItem
Item
Component
hasLimitOfMeasure
hasLink
hasKpi
hasIdea
hasAsset
hasProcess
hasProcessList
LimitOfMeasure
Link
KPI
Idea
Asset
Process
ProcessList
hasOutput
hasInput
Human
Assembly