Ref. Ares(2019)4124235 - 28/06/2019

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

MANUSQUARE

MANUfacturing ecoSystem of QUAlified Resources Exchange

D2.3
Area-specific models and inference rules

Dissemination Level	PU
Partners	HOLONIX, INESC, INNOVA, CSEM, SINTEF, SUPSI
Authors	Serena Albertario, Alessio Gugliotta, Marko Vujasinovic
Planned date of delivery	M18 – June 2019
Date of issue	30th June 2019
Document version	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 761145

DISCLAIMER:
The herewith information reflects only the author's view. The European Commission is not responsible for any use that may be made of the information herewith included.

DOCUMENT HISTORY

Version	Issue date	Content and changes	Author
0.1	18.06.2019	First version of deliverable D2.3	Serena Albertario
0.2	19.06.2019	Second version, after first review made by INESC	Serena Albertario, Henrique Diogo Silva
0.3	20.06.2019	Third version, after second review made by INNOVA	Serena Albertario, Marko Vujasinovic
0.4	25.06.2019	Final draft of the deliverable	Serena Albertario
1.0	28.06.2019	Quality assurance	Andrea Bettoni

Role	Partner	Person
Reviewer 1	INESC TEC	Henrique Diogo Silva
Reviewer 2	INNOVA	Marko Vujasinovic
Quality assurance	SUPSI	Andrea Bettoni

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	Description
B2B	Business to Business
CPM	Corporate Performance Management
ERP	Enterprise Resource Planning
GWP	Global Warming Potential
IPR	Intellectual Property Rights
IT	Information Technology
MANU-SQUARE	MANUfacturing ecoSystem of QUAlified Resource Exchange
PPS	Production Planning System
RPC	Remote Procedure Call
SME	Small and Medium Enterprises
WEF	World Economic Forum
WP	Work Package
FOAF	Friend of a friend, machine-readable ontology describing persons, their activities and their relations to other people and objects.
ACRT	Agent Certification Ontology

1 EXECUTIVE SUMMARY

This report addresses area specific models and inference rules behind the MANU-SQUARE (MANUfacturing ecoSystem of QUAlified Resource Exchange) platform. With area specific models it means tool-specific models and service-specifics models. Starting from the MANU-SQUARE Ecosystem's Core Data model and its ontological representation, defined in Task 2.1, an extension of the Core Data model has been done, taking into consideration tools' needs and business process that will be tested and validated through MANU-SQUARE project. This deliverable reports the extended version of the MANU-SQUARE Core data model, detailing the data model extension related to each specific tool of WP4. In addition to the model extension discussion, this document discusses inference rules support and gives few examples of the rules, which, however, will be further detailed and implemented in Task 2.4.

The outcome of this report is a data model for the MANU-SQUARE platform that conforms to both the defined business process as well as the tools requirements for all the functionalities implementation. A more detailed description of each specific tool data model will be reported in WP4 deliverables where the tools' development will be done.

The report consists of the following chapters:

- §2 introduces the aim and scope of the area-specific models for expansibility and reasoning inference rules.
- §3 describes the applied methodology which mainly consists of consortium partner meetings; analysis made on business process described in D5.1, analysis made on first version of the data model described in D2.1, extension of the data model for each specific tool.
- §4 outlines all the extension of data model for each specific tool of WP4.
- §5 describes the second version of MANU-SQUARE data model.
- §6 describes the first set of inference rules.
- §7 gives the reference to GITLAB: the repository used in the project to upload all documents related to technical activities.
- §8 concludes the report and presents the next steps.

2 INTRODUCTION

The MANU-SQUARE project creates an ecosystem that acts as a virtual marketplace bringing the available production capacity, as well as other virtual and physical assets, closer to the production demand to obtain the optimal matching (See Figure 1). This has two main advantages:

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

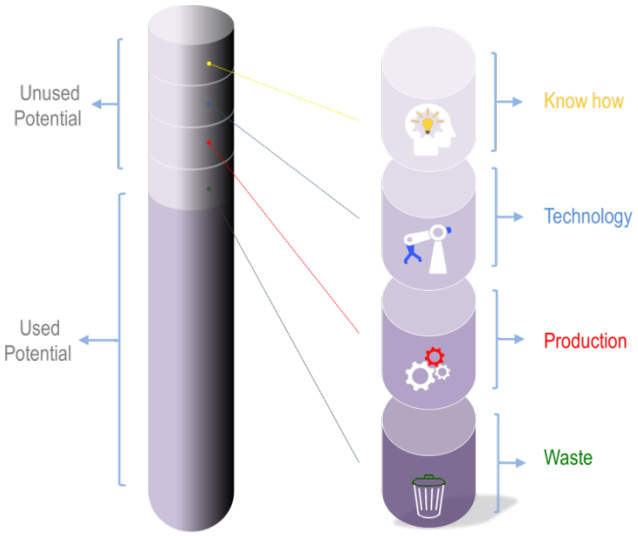

Figure 1 Composition of the unused potential

MANU-SQUARE establishes an ecosystem that is organized to match the needs of buyers with the availability of sellers in terms of know-how, technology, manufacturing capacity and waste. The associated MANU-SQUARE platform uses a Blockchain technology to ensure transparency and provide security, thereby fostering the building of trust amongst the different stakeholders of the platform. A manufacturer may have a role of a supplier (seller) or a customer (buyer). In the case of the latter, a manufacturer uses the platform to engage with the MANU-SQUARE ecosystem to fulfil a need, such as additional production capabilities. The platform performs the search for the optimal matching on a wide number of possible suppliers from the MANU-SQUARE ecosystem, using a sophisticated criterion that ensures high level of quality, reliability of suppliers, reduction of costs and short time to close the business transaction. The generated ecosystem allows optimal matches also for resources other than production hours or tangible assets with the aim to identify and exploit unexpected synergies between participants and to promote the mutual interaction of diverse industries, also within different value networks, for beneficial reuse of competences and flows.

The main objectives of the project are:

1. To make European unused manufacturing capacity emerge towards its reintegration in the loop and the creation of local efficient value networks.
2. To support innovative SMEs and start-ups in finding the optimal suppliers to transform their business ideas into new product-services.
3. To seamlessly involve actors all along the entire value network including consumers for cross-fertilization of product-service solutions and underlying technologies.
4. To coordinate the whole MANU-SQUARE ecosystem towards a better use of resources and a more sustainable European manufacturing.

2.1 Aim and scope of the area-specific models for expansibility and reasoning inference rules

Aim and scope of this document is the revision of the MANU-SQUARE platform data model. It represents the base for tools' and for architecture's platform development. Starting from the will to have data centralized and harmonized, this

work has been done to have a unified data model. Each MANU-SQUARE partner, owner of a specific tool, analysed business process, coming from task T5.1 and the first version of the data model, coming from Task T2.1. Considering tools' need, each partner extended the data model, inserting all required entities, attributes and connections related to its tool.

Once received all extended data models, it has been possible to design the MANU-SQUARE platform data model. It's the second version of the first one made at month M8 during Task T2.1. First version is included in Deliverable D2.1 "Semantic meta-model for ecosystem representation".

2.2 Relationships of T2.3 with other tasks

Task 2.3 is connected with tasks of WP1, WP3, WP4 and WP5. It takes inputs from Task 1.4, 5.1 and tasks from WP4 and WP3. Starting from the architecture (Task 1.4), it has been possible to have a first version of the data model in Task 2.1. All tasks of WP4 analysed tools' needs and in Task 5.1 the business' needs have been analysed. The result of Task 2.3 is the extended data model that will be developed during MANU-SQUARE platform development in WP3 and WP4. These relationships are depicted in Figure 2Figure 2 and explained below.

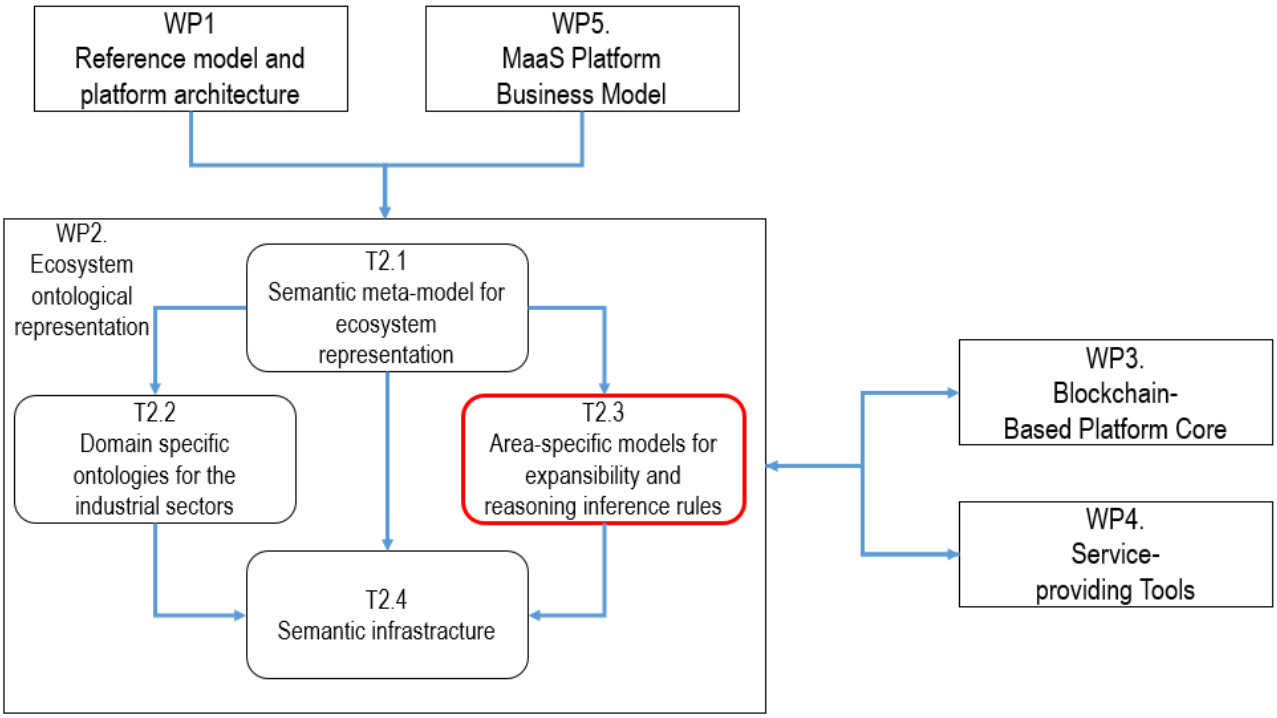

Figure 2 Relationship of Task 2.3 with other tasks

The interdependencies within WP2 are as follows:

- Task 2.1. The purpose of the task is to describe the core abstract concepts and main propositions of the semantic representation that constitutes the backbone structure of the ecosystem data model.
- Task 2.2. The purpose of the task is to expand the ecosystem core meta-model developed in Task 2.1, translating the domain-specific knowledge of the industrial sectors addressed in the project.
- Task 2.4. The purpose of the task is to design and develop the most suitable technical infrastracture to manage ecosystem data and knowledge in a uniform semantic representation, compatible with the models, ontologies and rules defined in previous tasks.

2.3 Outline

This report is organized into 7 chapters. The first chapter, §1, is an executive summary. §2 and §3 provide introduction, explain the aim and scope of the report, relationships and methodology applied. §4 presents the extension added to the

data model for each specific tool. The outcome, which is the MANU-SQUARE platform extended data model is reported in §5. §6 reports inference rules. The final chapter, §7, concludes the report.

3 METHODOLOGY

The extension of the data model defined in Task 2.1, is one of the most relevant activities required to build a platform that integrates all the necessary tools' requirements and business' needs.

The applied methodology to define the MANU-SQUARE platform data model is shown in Figure 3. It consists of analysis made by partners on the first version of the data model defined in Task 2.1, on business process' needs defined in Task 5.1 and in tools' needs of WP4. Partners involved have been: HOLONIX, IBM, INESC, INNOVA, SUPSI and SINTEF. They made first analysis and a match between results of these analysis. During physical and online meetings, all partners decided how to proceed. Each one extended the data model with the tool in his possession and then HOLONIX, as task's leader included all extended models in a final one, using tool Modelio.

Each data model extension has been also validated by all partner during the 3rd general assembly done in Milan on 20th and 21st of May.

After the definition of the extended data model, INNOVA has been able to define inference rules.

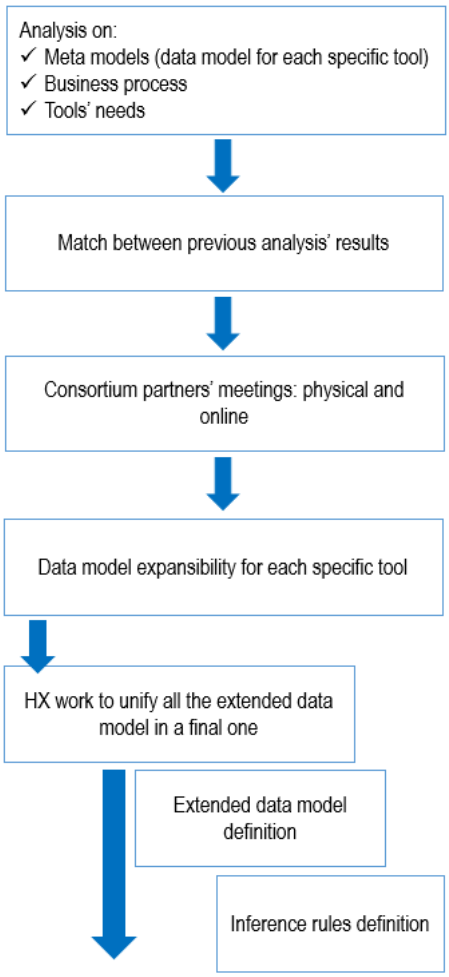

Figure 3 Extended data model definition methodology

3.1 Consortium partners meetings

Meetings with an involvement of consortium partners responsible for each specific tool have been done both physically and online. Main goals have been to define a methodology to extend data models, taking always into consideration business and tools' needs and receive updates of the work done by partners.

Table 1 shows all these meetings done with consortium partners.

Meetings	Topic	Partners involved
1st physical meeting 30th of October 2018 - Rome	1st presentation of Task 2.3 methodology and way to proceed	HOLONIX; IBM; INESC; INNOVA; SINTEF; SUPSI
2nd physical meeting 30th of November 2018 – Manno	Business process analysis, BPMN Model, T2.3: development roadmap definition	HOLONIX, SUPSI
3rd physical meeting 09th of January 2019 – Manno	General assembly, updates about the work done by partners in T2.3	HOLONIX; IBM; INESC; INNOVA; SINTEF; SUPSI
4th online meeting 08th of February 2019	Task 2.3 updates: check of work done by partners	HOLONIX; IBM; INESC; INNOVA; SINTEF; SUPSI
5th online meeting 13th of February 2019	Task 2.3 updates	HOLONIX; INNOVA
6th online meeting 6th of March 2019	Task 2.3 updates: check of work done by partners	HOLONIX; IBM; INESC; INNOVA; SINTEF; SUPSI
4th physical meeting 20th of May 2019 – Milan	General assembly, updates about the work done by partners in T2.3 and final validation of all extended data models	HOLONIX; IBM; INESC; INNOVA; SINTEF; SUPSI

Table 1 Consortium partners meetings recap

During meetings discussions about extensions have been conducted.

3.2 Template used to route the work in extended data models

To start the work a template has been created to help during the redaction of the extension of data model for each specific tool. The template is shown in Table 2. For each specific tool, partner inserted the Interlocutor, the role of their tool and the object. Object means the topic the tool is going to share with an other tool.

End (as Interlocutore)	Role	Object

Table 2: Template used to route data model extension

Considering DoA agreement, for tools and ownership, the template has been distributed to the partners for each specific component / tool as shown in Table 3.

Task	Tool / Software components	Owner
	Match making Tool	SINTEF
	User profiler and reputation mechanism	INESC
	Sustainability assessment layer	SUSPI
	Unified flow ecosystem orchestration	SUPSI
	Open Innovation, Co-Design, Idea Management Tool	HOLONIX

Table 3 Tools ownership

3.3 Business process analysis based on D5.1

Tools' owners made analysis on business process described in D5.1 to establish how to extend the data model in order to cover business' needs. Figure 4 and Figure 5 show the business process took into consideration. To have more details about them, please refer to Deliverable D5.1.

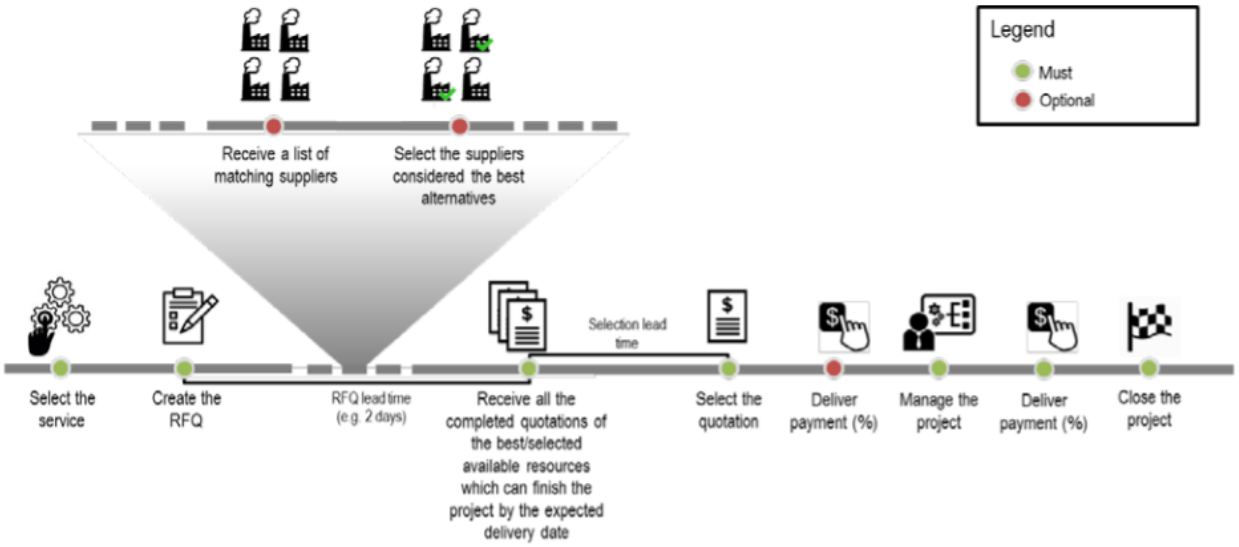

Figure 4: Resource sharing service: touch points with the platform from customer's perspective

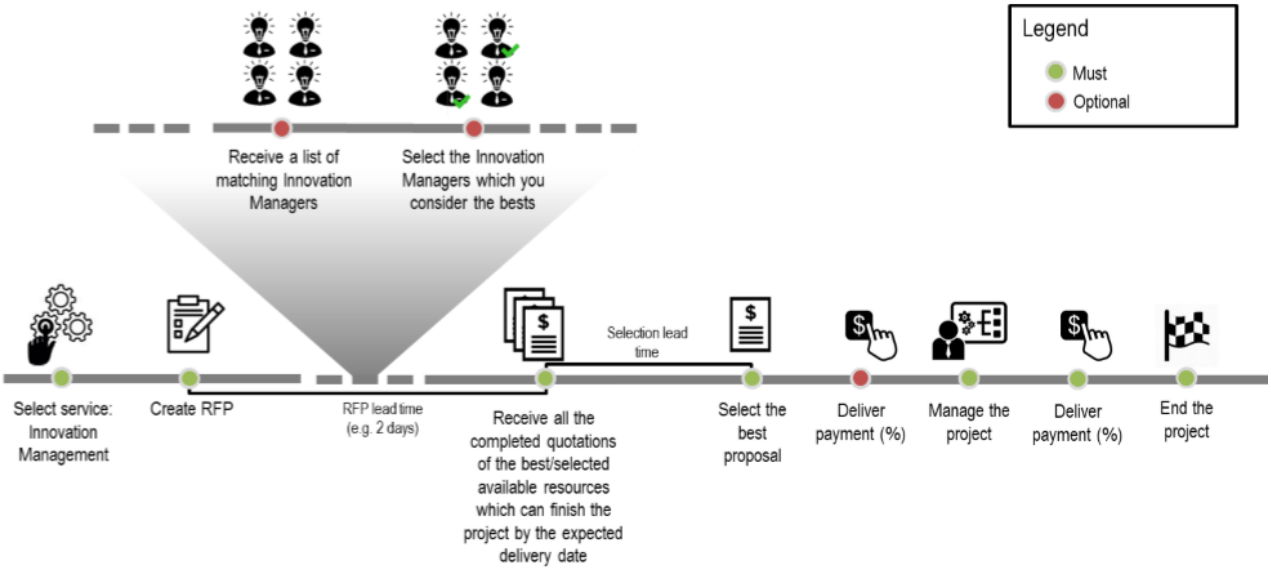

Figure 5: Innovation Management service: touch points with the platform from customer perspective

3.4 Data model analysis based on D2.1

Data model extension started from the ecosystem core data model, provided in Deliverable D2.1. Each partner, tools' owner, analyzed the core data model and inserted new entities, attributes and field in order to cover the specific needs of its tool. For more details about the first version of the ecosystem core data model, please refer to Deliverable D2.1.

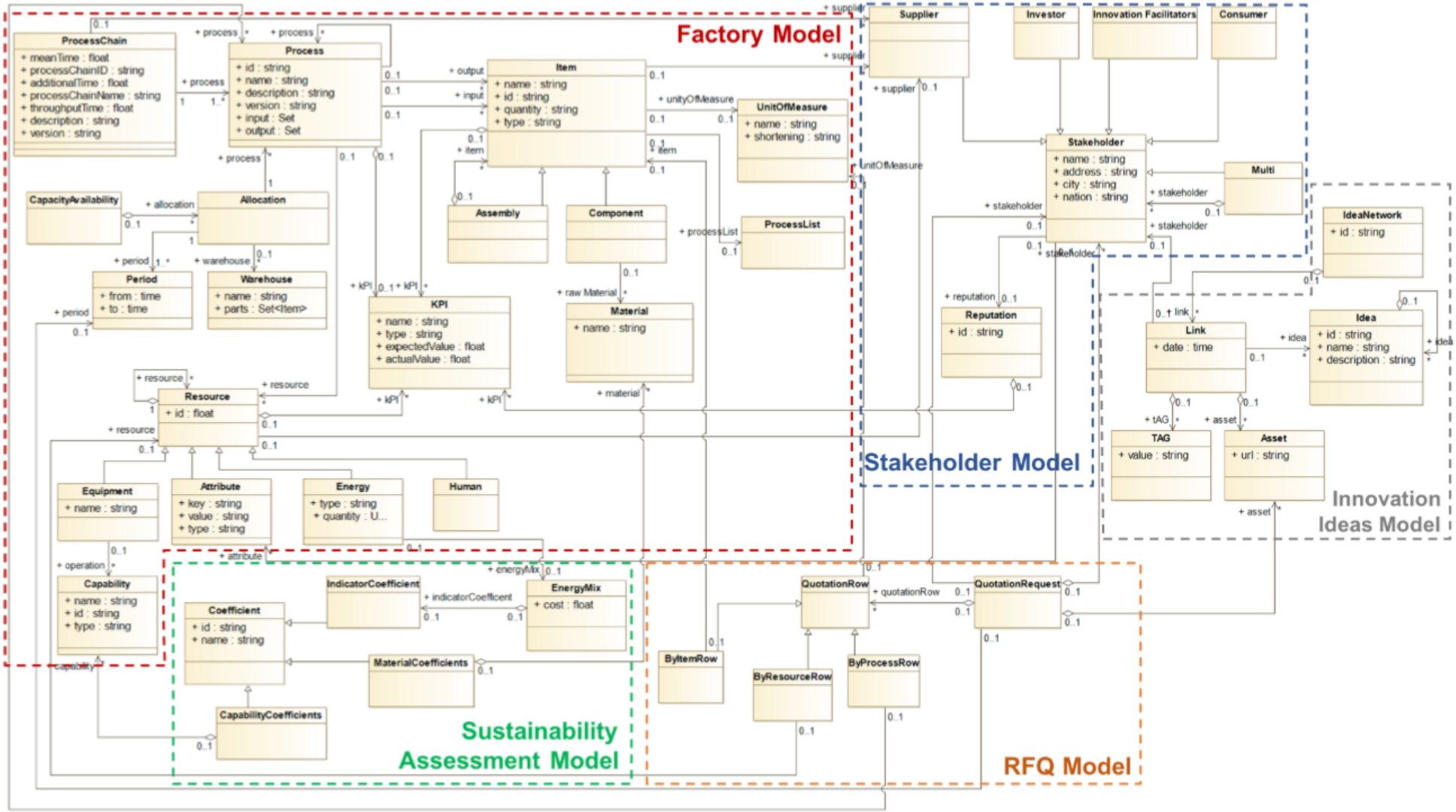

Figure 6: First version of MANU-SQUARE platform data model

4 EXTENSION OF DATA MODEL FOR EACH SPECIFIC TOOL

Starting from the MANU-SQUARE platform architecture, defined in Task1.4 and represented in Figure 7, it has been possible to define which tool had to extend the data model.

In tools' layer, these tools have been highlighted and subjected to the data model extension:

- Match making tool
- User profiler and reputation mechanism
- Sustainability Assessment Layer
- Unified Flow Ecosystem Orchestrator
- Open Innovation, co-design idea Mngt Tool

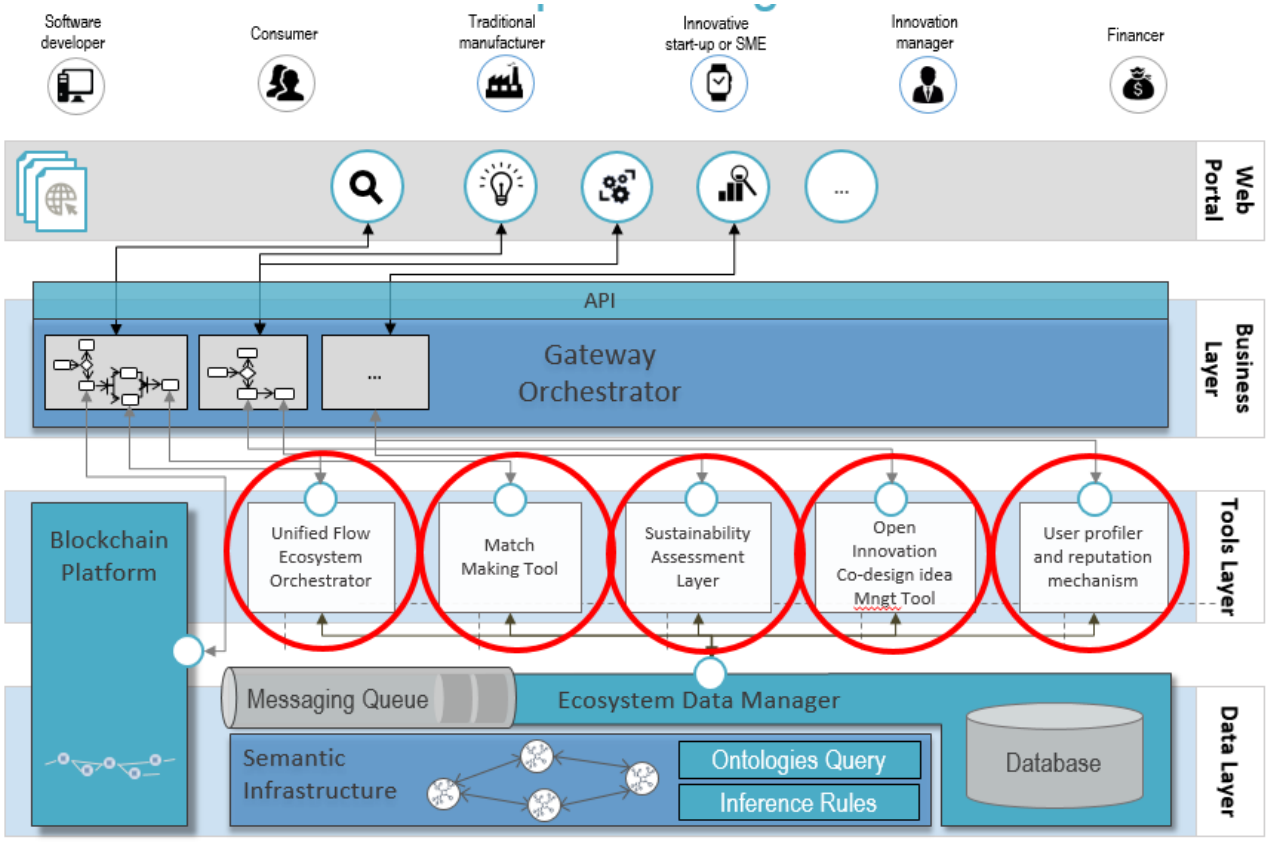

Figure 7 MANU-SQUARE platform architecture

Chapters here below will show how each partner extended the data model, considering his tools' needs and business process' needs.

4.1 Match making tool

Based on the first version of the Ecosystem Core data model and its ontological representation, SINTEF has done initial matchmaking algorithm trials and competency questions execution (by means of simple SPARQL queries) to assess the coverage and scope of the MANU-SQUARE ontology. Based on the trials and findings, SINTEF declared at Month 18 that the matchmaking tool has no need for an extension of the current data-model and to it related ontology. However, small changes in the model, thus, ontology, could come in next months, depending on what will be the optimal matchmaking strategy (e.g. tool could store some previously made matchmaking results and computations to gain efficiency for new matchmaking requests.).

4.2 User profiler and reputation mechanism

Figure 8 represents the extension made by INESC to the first version of the data model regarding user profiler and reputation mechanism tool.

Here below a summary of changes brought by INESC to the first version of data model:

1. More attributes to the main Stakeholder class that allow for a better base characterization of users;

2. Introduction of two new abstractions to the Stakeholder class: Organization & Person.

2.1. Organization: Equivalent to the FOAF class with the same name, this abstraction not only supports the interoperability of the ontology, but also allows for the existence of an organization as a user of the platform.

3. Introduction of a new grouping abstraction to the Stakeholder class: Group. This abstraction takes the aggregations made possible by the class Multi to a level above the stakeholder. It serves to represent relevant named groupings of different Stakeholders.

4. Introduction of the Certification class. This new class, based on the ACRT ontology, makes possible to have associated with stakeholders different types of certifications, along with proof of this certification. This class was designed with the notion that an entity of the Stakeholder class can be both certified and certify other entities (but not itself).

5. Addition of the *knows* property that introduces the concept of an I-know-you relationship between stakeholders. Although this feature requires more discussion within the consortium for its application, this information may prove valuable, as an example, when taken into account by the matchmaking algorithm in order to not return already known suppliers for a given query. Interestingly, knows relationship between stakeholders does not have to assert into the platform, but with a support of semantic infrastructure can be inferred from the data of business processes established thru the platform.

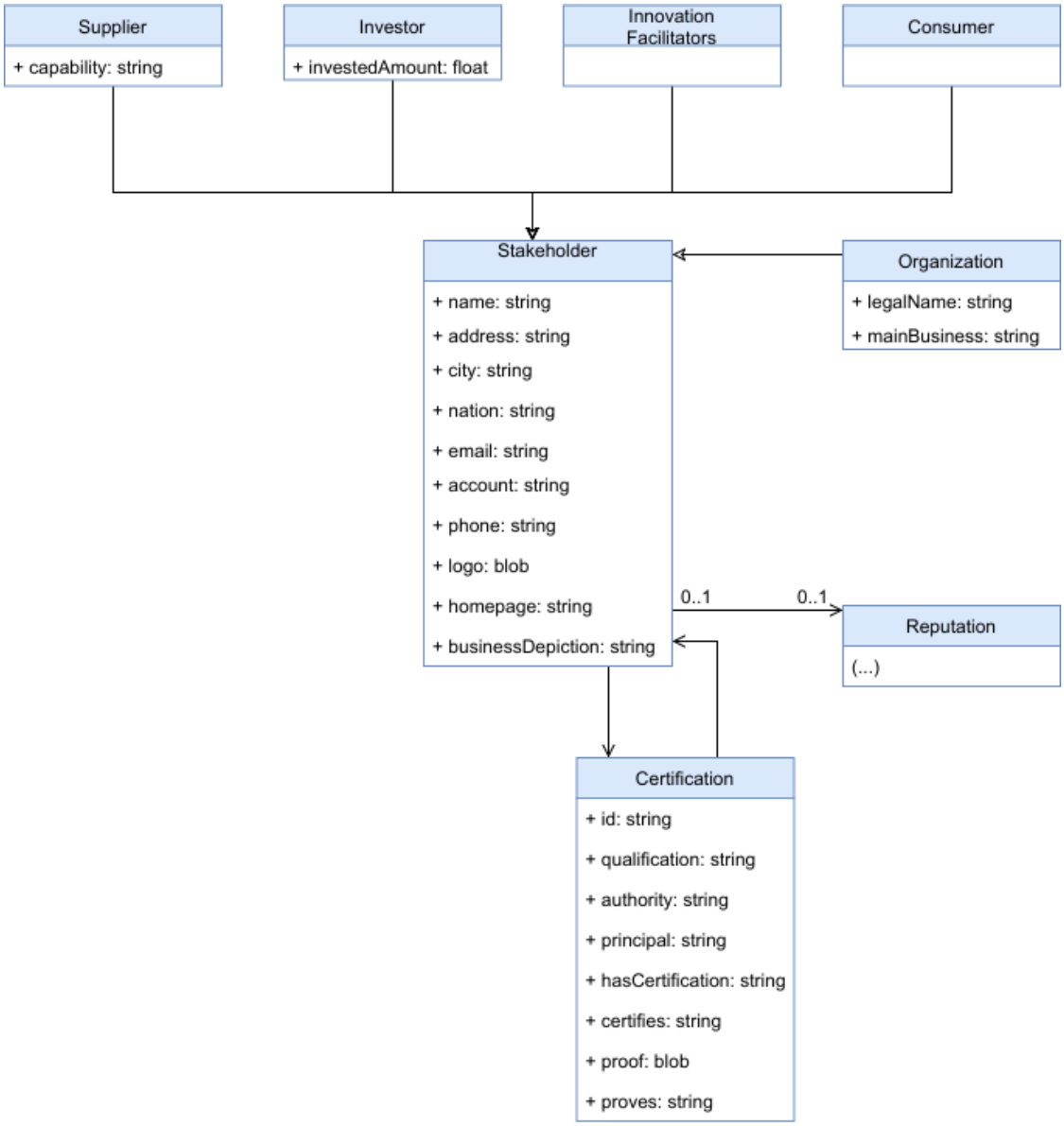

Figure 8: User profiler and reputation mechanism extension

Class descriptions are here detailed:

Stakeholder

Added attributes to better depict Stakeholders.

Field	Type	Description
Id	String	Identification of the element.
Name	String	A reference that can identify the represented subject.
BusinessDepiction	String	Description of the Stakeholder's main area of business
Address	String	Information about stakeholders contacts.
City	String	

Nation	String	
Email	String	
Phone	integer	
Account	String	Stakeholder external accounts.
Homepage	String	Stakeholder homepage. A homepage is a public Web document that is usually con- trolled, edited or published by the thing whose homepage it is; as such one might look to a homepage for information on its owner from its owner
Logo	Blob	A logo representing the Stakeholder.

Table 4: Stakeholders' added attributes

Certification

The Certification class is responsible for abstracting the certifications that stakeholders both have or certify. This general class is designed to be able to handle all the necessary information for describing a certification, as well as serve as prove it's validity. It's only relationship is with the Stakeholder class as only stakeholders have, and provide certifications.

Field	Type	Description
Id	String	Identification of the element.
Qualification	String	Description of what achievement, skill, or fact is being certified.
Authority	String	Which Stakeholder (or what) is doing the certifying.
Principal	String	Who or what stakeholder is being certified.
hasCertification	String	Specifies that a stakeholder possesses a given certification.
Certifies	String	Specifies that its subject, a stakeholder, has issued a certification.
Proof	Blob	Points to some proof of the certification
Proves	String	Asserts that its subject is somehow proof of a given certification

Table 5: Certification added attributes

Organization

Specializes the Stakeholder class, corresponding to social institutions such as companies, societies etc.

Field	Type	Description
Id	String	Identification of the element.
Legal name	String	Legal name of an organization
Main business	String	Main business of an organization

Table 6: Organization added attributes

Group

Represents a named collection of individual stakeholders.

Field	Type	Description
Id	String	Identification of the element.
Name	String	Group name

Table 7: Group added attributes

4.3 Sustainability assessment layer

Figure 9 represents the extension made by SUPSI to the first version of the data model regarding sustainability assessment layer tool.

The first version of the core data model already covered the major part of the needs of the Sustainability Assessment tools, except for some small changes: the process formalization has been updated in order to allow the description of a process according to a graph prospective, for that reason the concept of *ProcessArc* has been introduced. Thanks to this new entity, the output of a process can be formalized as input for another process (for example: the ouput of the electricity supply process becomes the input the injection moulding process).

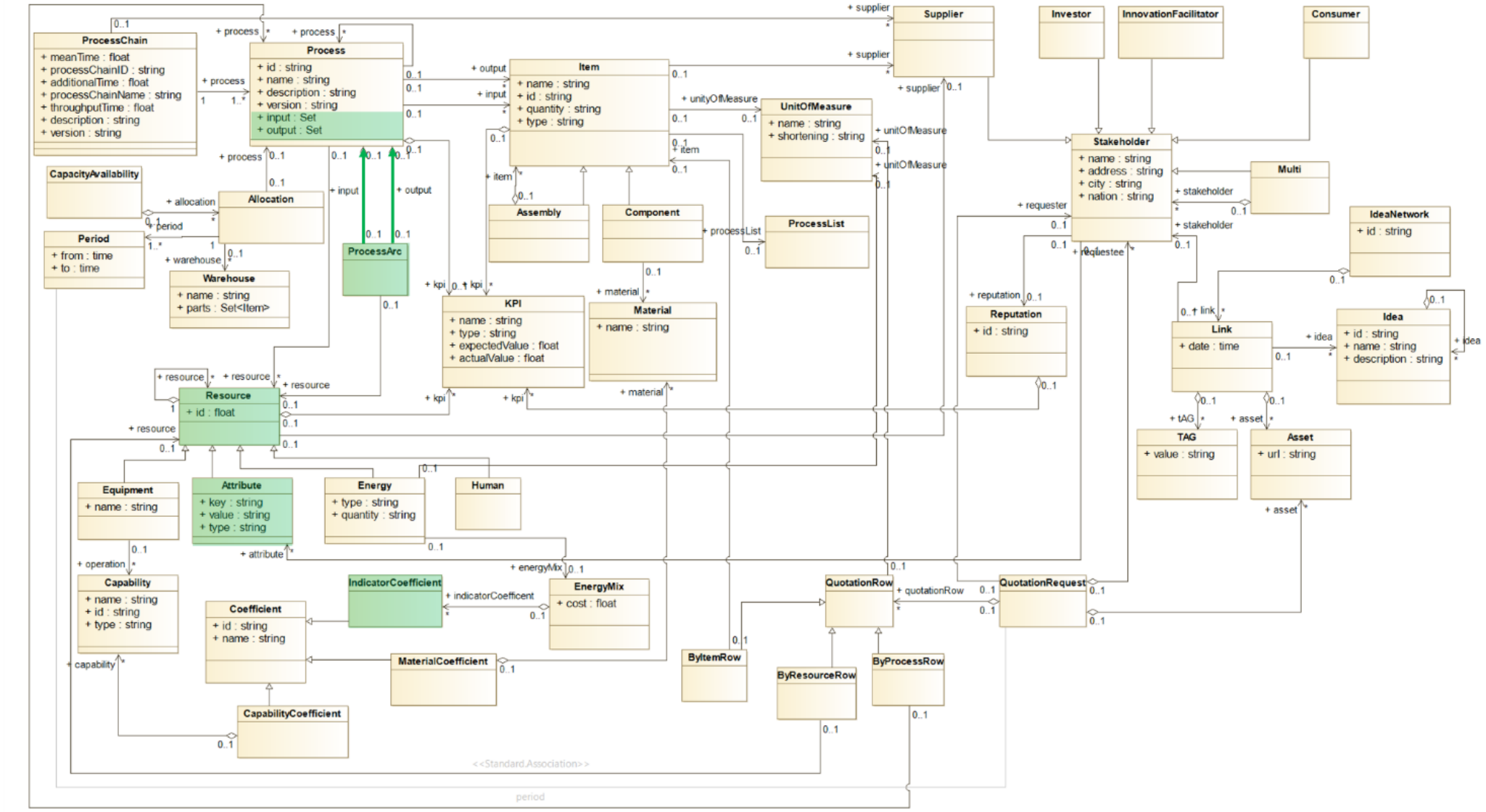

Figure 9: Sustainability assessment layer extension

The structure that allows to organize the processes in graph structures has been introduced in order to include in the datamodel also data coming from third parties, in particular from Ecoinvent (https://www.ecoinvent.org). Ecoinvent provides life cycle inventory data to describe the flows of a system product from and to nature. It describes each process as a series of input and output flows. Each input or output flow is also described in the Ecoinvent repository as a process.

4.4 Unified flow ecosystem orchestration

Figure 10 represents the extension made by SUPSI to the first version of the data model regarding the unified flow ecosystem orchestration tool.

The part of the data model dedicated to support the *RfQ* process has been adapted in order to consider the concept of *Project*. A *Project* is composed by a set of *Assets* (intended as a set of attachments) and can be a *RfP* or *RfQ* project. In particular, a *RfQ* process has 2 sets of *Attributes* meant to describe the aspects of the supplier and the processes which the customer is requesting. A *Supplier*, which is interested to a specific RfQ, has to sign an *NDA* in order to submit a *Quotation*. If the quotation will be accepted by the *Customer*, a new *Contract* will be created.

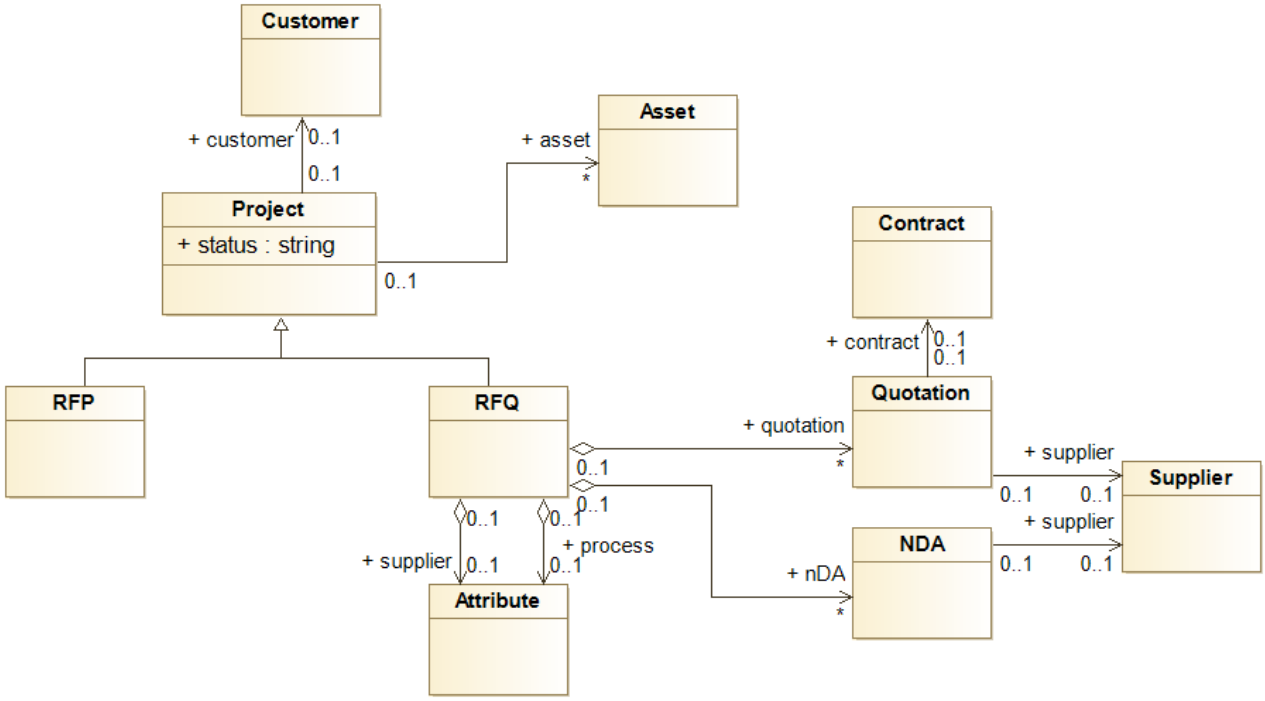

Figure 10: Unified flow ecosystem orchestration extension

4.5 Open Innovation, co-design idea Mngt Tool

In order to define all entities and attributes required to extend the data model for IDEA MANAGER's tool, HOLONIX took into consideration the process that will be implemented through IDEA MANAGER usage (Figure 11).

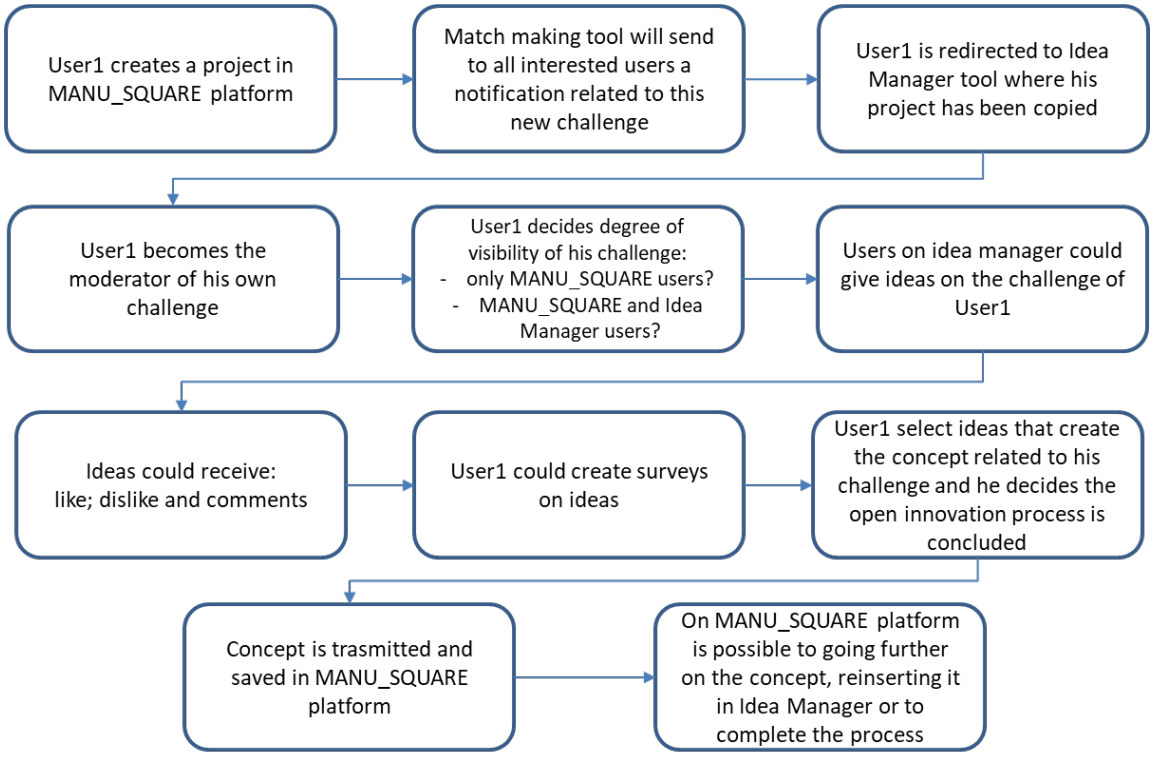

Figure 11: IDEA MANAGER process

Figure 12 represents the extension made by HOLONIX to the first version of the data model regarding the open innovation, co-design idea management tool.

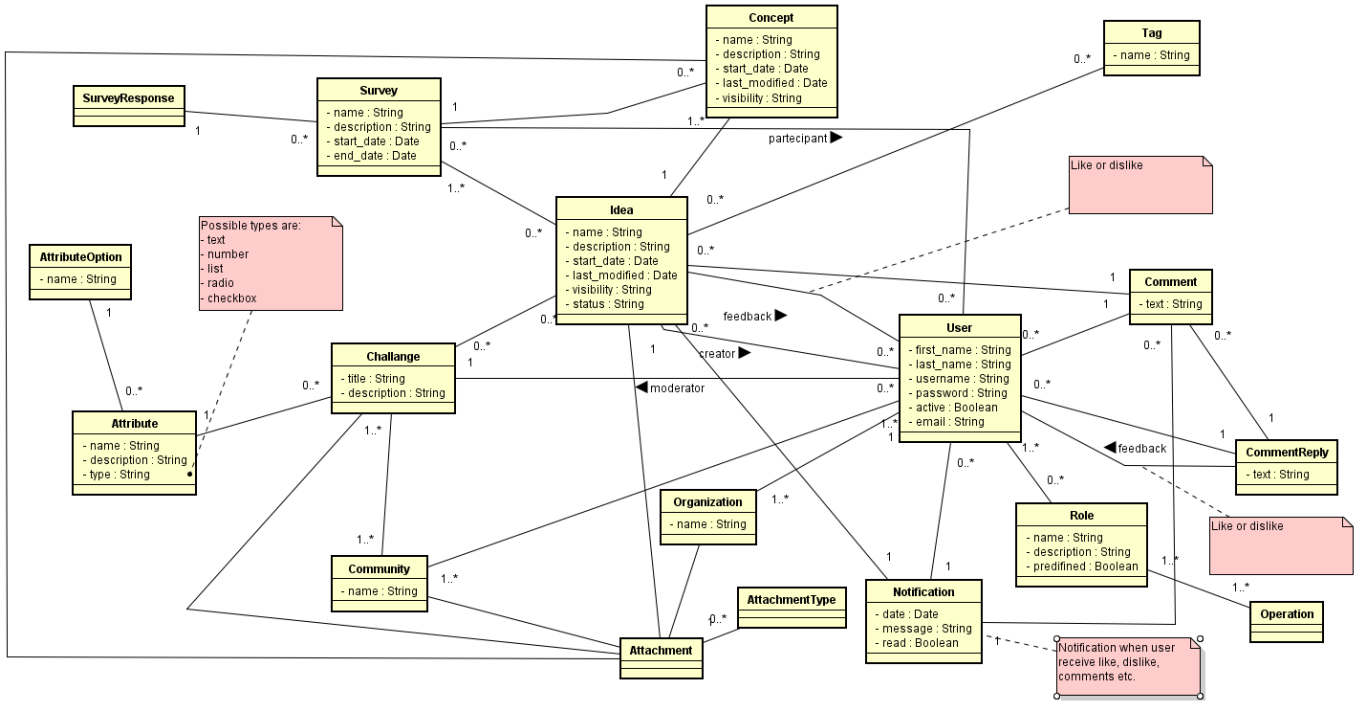

Figure 12: Open innovation, co-design idea mngt tool extension

Starting from the entity of the data model "Innovation ideas model", Holonix introduced:

Field	Type	Description

Concept	String	With a name, a description, a start date, a last modified date and a degree of visibility, it represents the result of the process implemented on Idea Manager tool
Challenge	String	With a title and a description, the user will insert in the MANU-SQUARE platform his challenge
Survey	String	It represents all the answers to the survey
Community	String	On Idea Manager there will be two communities: the Idea Manager and the Manu-square one. User will decide the visibility degree of his challenge choosing the community
Organization	String	Each user will belong to an organization
User	String	User will be characterized by name, surname, username, password, active, email. All these information will be included during the authentication phase
Role	String	Each user will have a specific role, roles will be predefined
Notification	String	A notification will be sent to the user once his challenge will receive comments, like, dislike, etc
Attachment	Blob	It will be possible to insert attachment to an idea, an organization, etc.
Comment	String	This string represents comments that challenge and ideas could receive
Comment reply	String	This string represents comments that other comments could receive

5 SECOND VERSION OF MANU-SQUARE DATA MODEL

This chapter represents the work done following each tool extension.

Using software Modelio, Holonix made a second version of the MANU-SQUARE platform data model, including all previous ones, ensuring consistency of the result.

5.1 First version, from D2.1

Figure 13 represents the first version of the MANU-SQUARE platform data model, including in Deliverable D2.1.

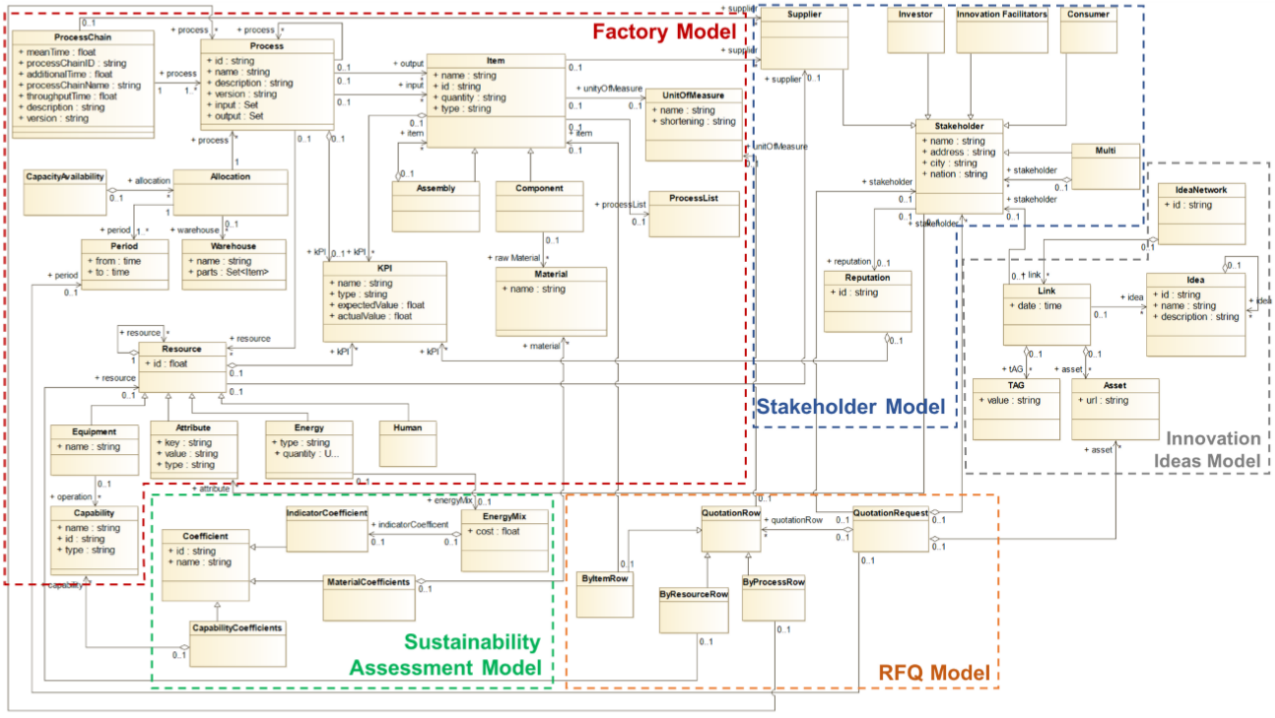

Figure 13: First version of MANU-SQUARE platform data model

5.2 Second version at M18

Figure 14 represents the second version of the MANU-SQUARE platform data model, taking into consideration all extensions made by all partners for each specific tool.

The integration of all extended data model has been done by Holonix using Modelio tool.

In Figure 14 only classes are included, while in next figures the details of each sections are shown with their attributes. In next figures, blue lines highlight limits' sections and outside these blue lines all classes of close sections are included, if there's a link in the total data model.

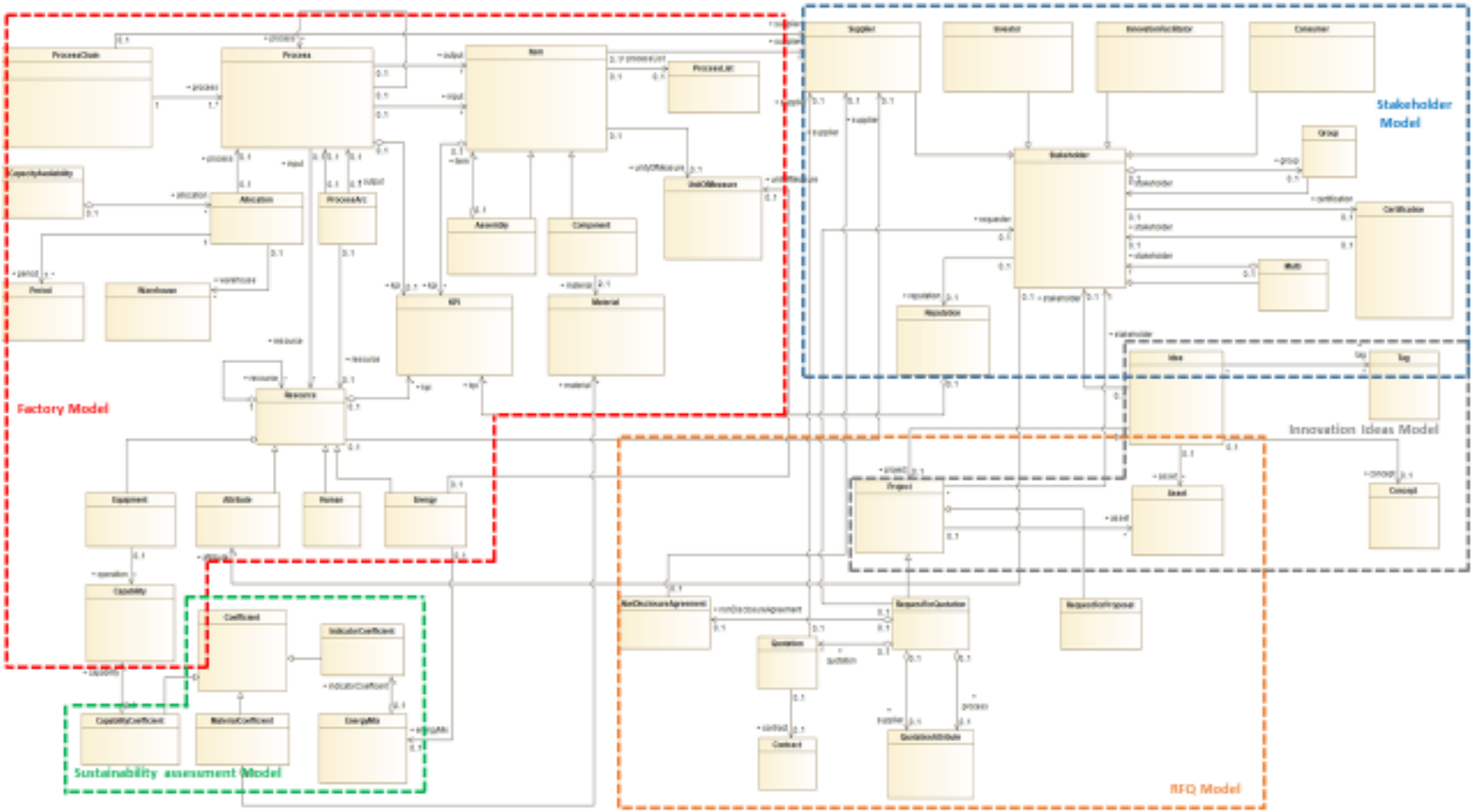

Figure 14: Second version of MANU-SQUARE platform data model

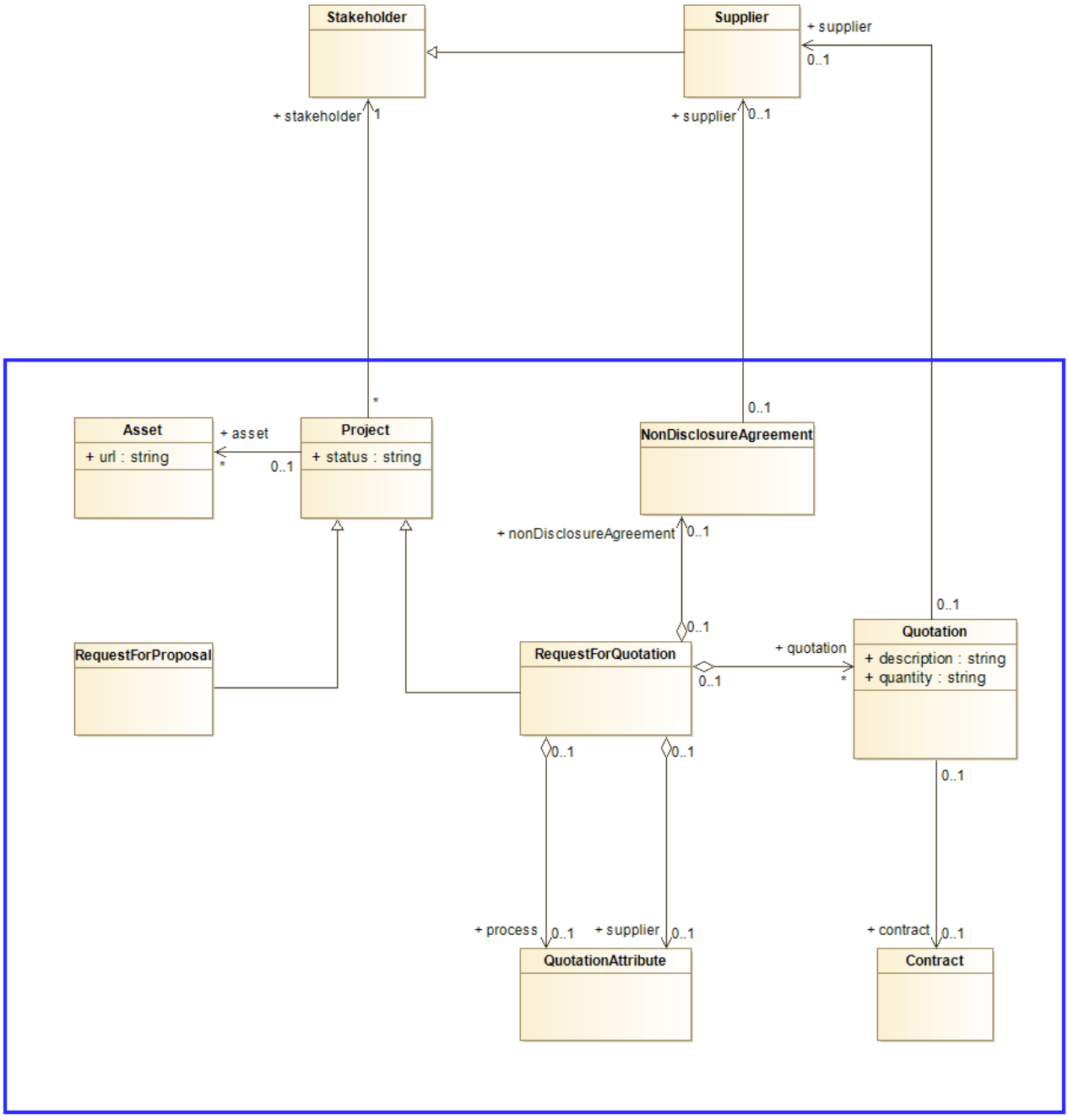

Figure 15: RFQ Model

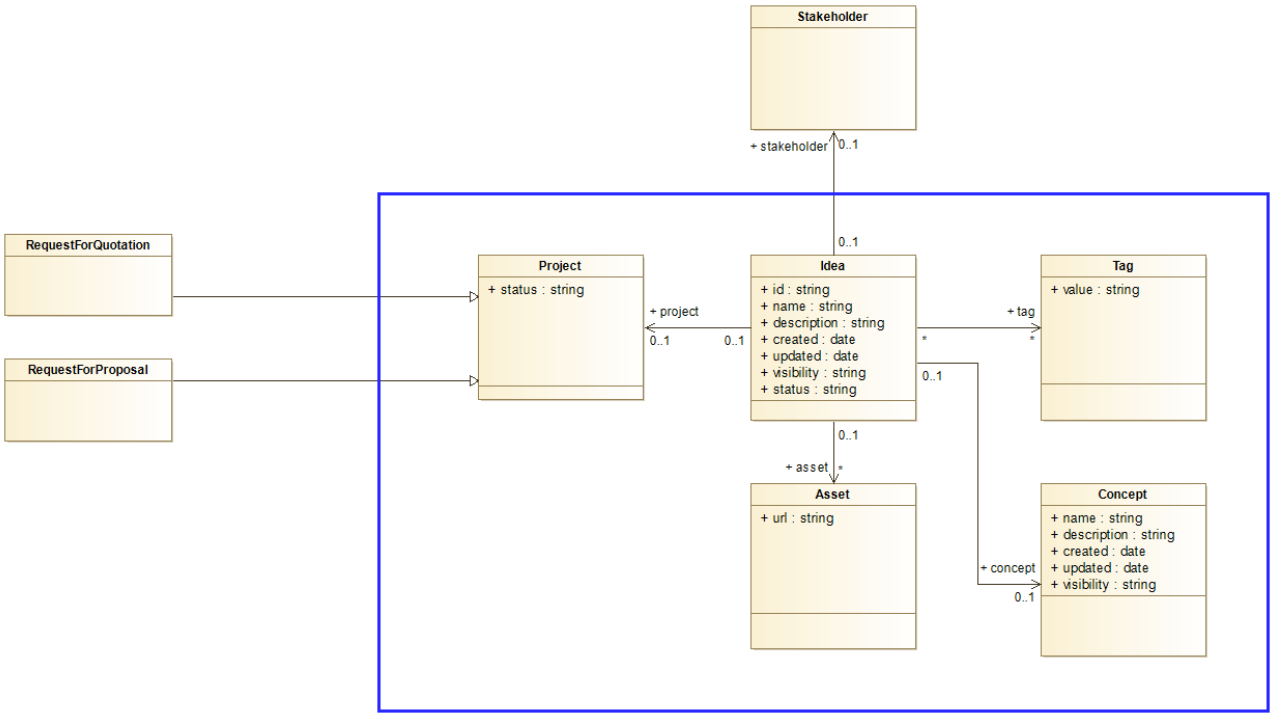

Figure 16: Innovation Ideas model

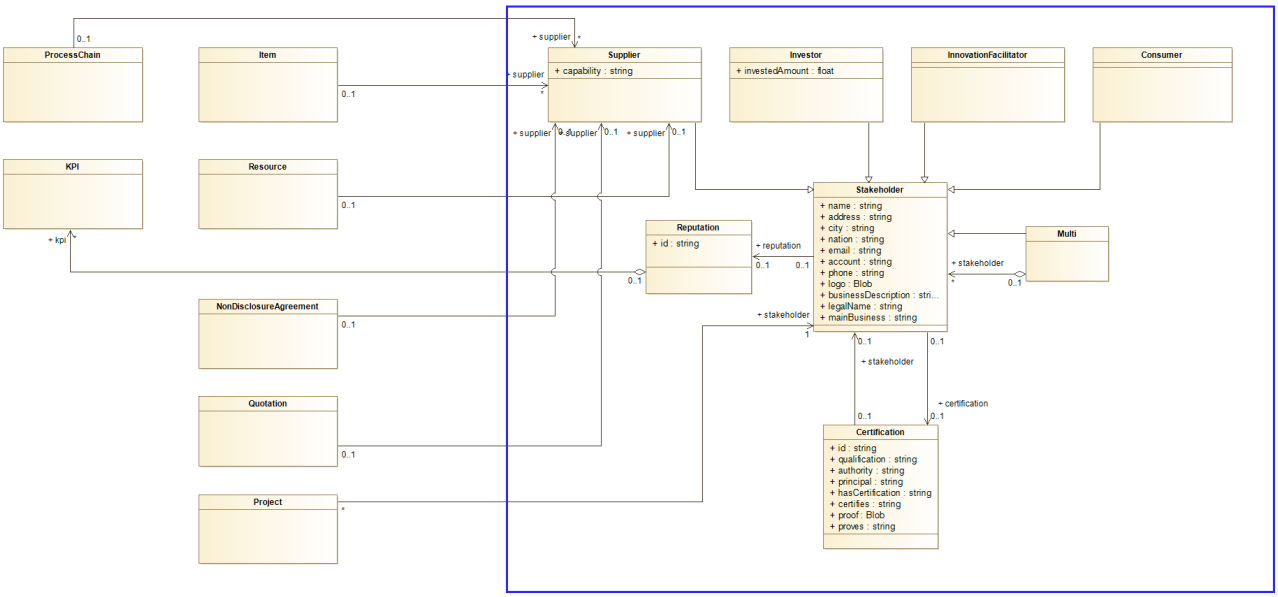

Figure 17: Stakeholder model

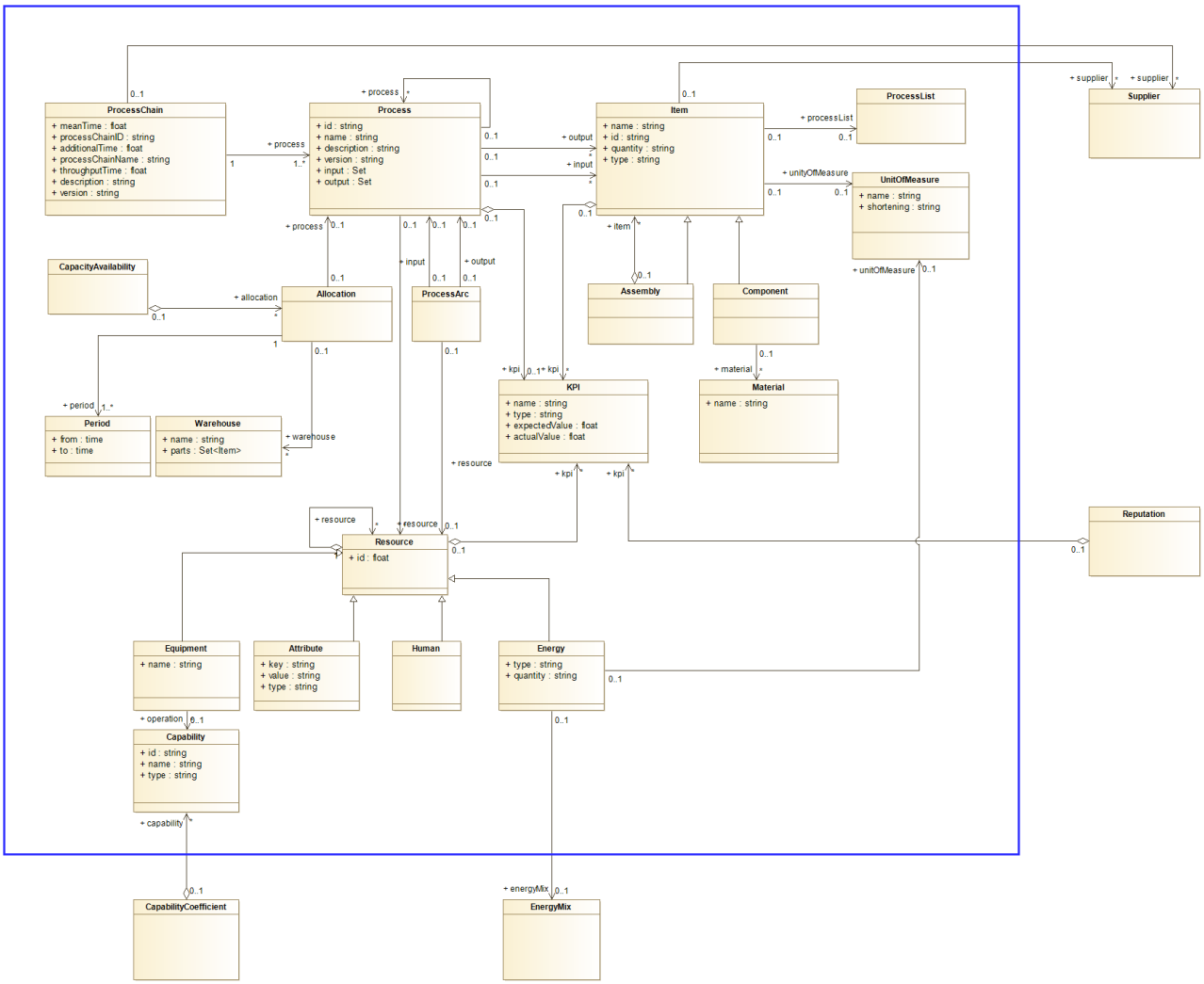

Figure 18: Factory model

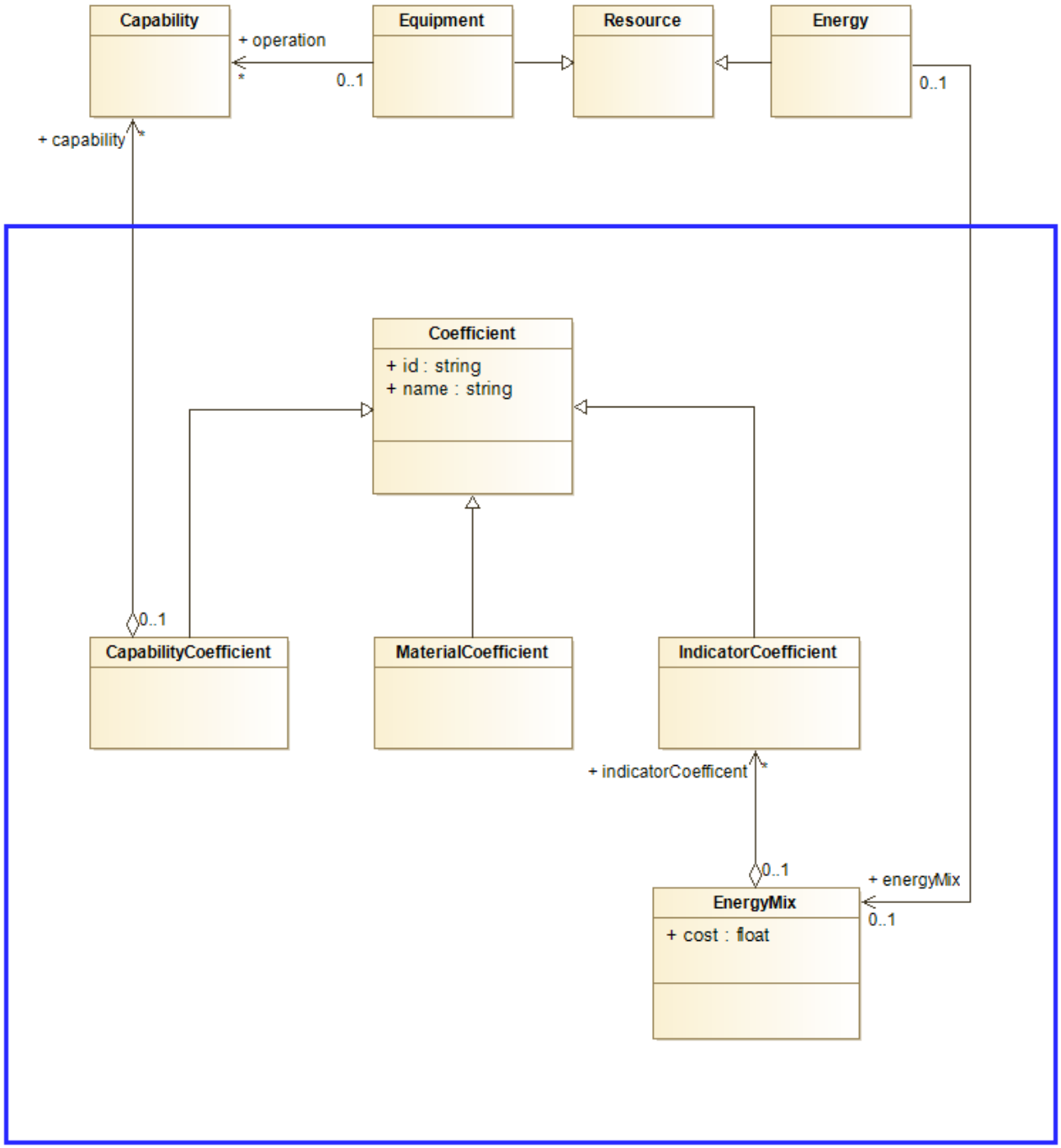

Figure 19: Sustainability data model

6 INFERENCE RULES

Besides the fine-tuning and improvements of the MANU-SQURE ecosystem ontology introduced in the previous sections, Task 2.3 was in charge of identifying the types of inference rules that can to be supported by the Semantic Infrastructure.

In this chapter, a first set of inference rules that will be implemented in Task 2.4 is described. They allow to find and infer new facts by reasoning of the data present in the ecosystem representations. These findings are for example the basis for proposing to the platform users suggestions about new opportunities residing in the ecosystem or providing more comprehensive descriptions of their resources, capabilities and products.

Depending on what MANU-SQUARE needs are, different type of rules can be supported by its semantic infrastructure. Specifically, within the current setup of the semantic infrastructure, the following types of rules would be supported:

- SPIN Inference rules
- RDFS semantics inference
- Jena inference rules
- Inference with SPARQL CONSTRUCT

These four types of rule support differ in several aspects, including syntax and execution algorithms/engines, however, can complement each other's to deliver a powerful inference system for the MANU-SQUARE ecosystem.

Rule type	Execution time	Supported by GraphDb	Allows custom rules	Rule chains and execution ordering
SPIN Inference Rules	Real-time, automatically applied to new asserted instances of the class associated with a rule and its subclasses.	Yes	Yes	Yes, through spin:nextRuleProperty
RDFS Reasoning	Real-time i.e., automatically applied after data is inserted into repository	Yes	No	No
Jena rules	On-demand (before or after inserting data into repository)	No	Yes	Yes, natively supported by forward chaining algorithm
Inference with SPARQL CONSTRUCT	On-demand (before or after inserting data into repository)	Yes	Yes	No, unless custom meta-data attached to rules, and engine implemented to use those meta-data

Table 8: Rules' type

6.1 SPIN Inference Rules

SPIN (https://spinrdf.org/) is a way to represent a wide range of business rules based on SPARQL expressions. In fact, SPIN is also referred to as *SPARQL Rules*. SPARQL is a well-established W3C standard implemented by many industrial-strength RDF APIs and all databases. This means that rules can run directly on RDF data without a need for materialization.

SPIN combines concepts from object oriented languages, query languages, and rule-based systems to describe object behavior on the web of data. One of the key ideas of SPIN is to link class definitions with SPARQL queries to capture rules and constraints that formalize the expected behavior of those classes. To do so, SPIN defines a light-weight collection of RDF properties and applies the associated rules to all instances of class, in a real time.

These rules are implemented USING SPARQL CONSTRUCT or SPARQL UPDATE requests (INSERT and DELETE) and can be used for example:

- **Calculate the value of a property based on other properties** – to give an idea: the area of a rectangle as a product of its length and width.
- **Isolate a set of rules to be executed under certain conditions** – in particular, to support incremental reasoning, to initialize certain values when a resource is first created, or to drive interactive applications.

Example[1]: Inference Rules using CONSTRUCT

If the property spin:rule points to a CONSTRUCT query, then this defines an inference rule that defines how additional triples can be inferred from what is stated in the WHERE clause. For each binding of the pattern in the WHERE clause of the rule, the triple templates from the CONSTRUCT clause are instantiated and added as inferred triples to the underlying model. At query execution time, the SPARQL variable ?this is bound to the current instance of the class.

The following example defines a rule that infers the values of the ex:grandParent property from values of ex:child.

```
ex:Person
      a       rdfs:Class ;
      rdfs:label "Person"^^xsd:string ;
      rdfs:subClassOf owl:Thing ;
      spin:rule
              [ a       sp:Construct ;
                sp:text """
                    CONSTRUCT {
                        ?this ex:grandParent ?grandParent .
                    }
                    WHERE {
                        ?parent ex:child ?this .
                        ?grandParent ex:child ?parent .
                    }"""
              ] .
```

.

Another common need in applications is **to check validity of the data**. For example, you may want to require that a field is entered and/or that the string entered follows your format requirements. SPIN offers a way to do constraint checking with closed world semantics and automatically raise inconsistency flags when currently available information does not fit the specified integrity constraints. Constraints are specified using SPARQL ASK or CONSTRUCT queries, or corresponding SPIN Templates.

The following example constructs constraint violations whenever a person's spouse has the same gender as the current person.

```
spin:constraint
          [ a       sp:Construct ;
            sp:text """
                CONSTRUCT {
                    _:violation a spin:ConstraintViolation ;
                         spin:violationRoot ?this ;
                         spin:violationPath kennedys:spouse ;
                         spin:violationValue ?spouse ;
                         spin:violationLevel spin:Warning ;
                         rdfs:label "Same-sex marriage not permitted (in this model)"
                }
                WHERE {
```

[1] https://spinrdf.org/spin.html#spin-rules-construct

```
            ?this kennedys:spouse ?spouse .
            ?this kennedys:gender ?gender .
            ?spouse kennedys:gender ?spouseGender .
            FILTER (?gender = ?spouseGender) .
        }"""
    ] .
```

6.2 RDFS/OWL reasoning and custom rules in GraphDb

MANU-SQUARE's semantic infrastructure is built on a GraphDB triple store. GraphDB performs reasoning based on forward-chaining of RDFS and OWL entailment rules. It applies a total materialisation reasoning strategy, where the inference rules are applied repeatedly to the asserted (explicit) statements until no further inferred (implicit) statements are produced. GraphDb provide several predefined rulesets (Table below), which can be used as-is, or further customized with semantics better tuned to the particular domain.

Table 9 (source http://graphdb.ontotext.com/documentation/standard/reasoning.html)

Ruleset	Description
empty	No reasoning, i.e., GraphDB operates as a plain RDF store.
rdfs	Supports the standard model-theoretic RDFS semantics. This includes support for subClassOf and related type inference, as well as subPropertyOf.
rdfs plus	Extended version of RDFS with the support also symmetric, inverse and transitive properties, via the OWL vocabulary: owl:SymmetricProperty, owl:inverseOf and owl:TransitiveProperty.
owl-max	RDFS and that part of OWL Lite that can be captured in rules (deriving functional and inverse functional properties, all-different, subclass by union/enumeration; min/max cardinality constraints, etc.).
owl2-ql	The OWL2 QL profile - a fragment of OWL2 Full designed so that sound and complete query answering is LOGSPACE with respect to the size of the data. This OWL2 profile is based on DL-LiteR, a variant of DL-Lite that does not require the unique name assumption.
owl2-rl	The OWL2 RL profile - an expressive fragment of OWL2 Full that is amenable for implementation on rule engines.

MANU-SQUARE does not need yet the complexity of the most expressive supported semantics (owl-ones), and we opted to choose one of the less complex, which will result in faster inference. So far gathered requirements for the platform can be addressed with an entailment of RDFS semantics of subClassOf and subPropertyOf relationships and classification based on domain and range definitions of RDF properties. SubClassOf is used as a main construct to structure domain-specific taxonomies of MANU-SQUARE core concepts, hence, the query answering in MANU-SQUARE can give more results based on inferred classifications of instances. OWL reasoning is not foreseen in MANU-SQUARE at this stage, however, if new requirements emerge, a GraphDB instance can be re-configured to use one of the OWL's rulesets.

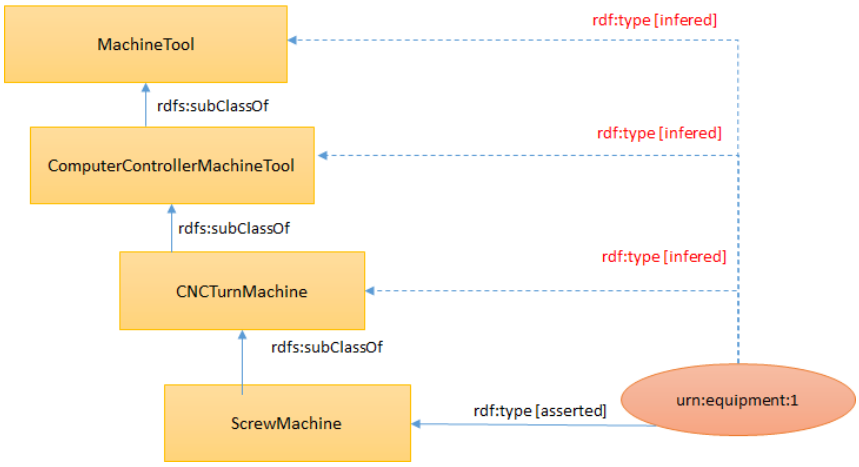

Figure 20 RDFS subClassOf inference

In GraphDb, RDFS inference is achieved via a set of axiomatic triples (RDF Schema statements) and RDFS entailment rules to allow the full set of valid inferences using RDFS semantics. In addition to RDFS entailment rules, a ruleset can be customized with semantics that is better tuned to the particular domain or application requests. For example, the listing in below shows two rules for RDFS subClassOf inference and one custom rule for MANUSQUARE domain-specific inference.

```
Id: rdfs1
a <rdf:type> b
b <rdfs:subClassOf> c [Constraint b != c]
-----------------------------------
a <rdf:type> c


Id: rdfs2
a <rdfs:subClassOf> b [Constraint a != b]
b <rdfs:subClassOf> c [Constraint a != c, b != c]
-----------------------------------
a <rdfs:subClassOf> c


Id: manusquare-custom1 (if some resource hasItem y, then type of x is an Assemby)
x  <core:hasItem>  y
------------------------------
x  <rdf:type>  <core:Assembly>
```

6.3 Rules with SPARQL Construct

As introduced in Section 6.1, the SPIN representation adopts SPARQL Construct capabilities to link a rule to a specific ontology class and trigger inference when a new instance of such a class is created (or updated). In a similar fashion, the SPARQL Construct can be used to create more general rules that are not necessarily linked to a class.

If we view a CONSTRUCT statement as the definition of a rule and the resulting new triples as the result of the execution of the rule, then we have a rules language.

For example, the following SPARQL "rule" says that if ?person1 has the spouse ?person2 and the home telephone number ?phoneNum, then ?person2 also has the home telephone number ?phoneNum.

```
CONSTRUCT { ?person2 v:homeTel ?phoneNum . }
WHERE {
  ?person1 :spouse   ?person2 ;
           v:homeTel ?phoneNum .
}
```

6.4 In-memory reasoning by Jena rules applied on preInsert and postInsert events for named graphs

Jena is an open source Semantic Web framework for Java (https://jena.apache.org/), with an I/O API for RDF graphs and OWL ontologies, with in memory and persistent storage for RDF triples, and inference subsystem that offers a range of reasoners for deriving additional facts. Jena inference subsystem includes Transitive reasoner which implements transitive and reflexive properties, RDFS rules reasoner containing RDFS entailments, OWL reasoner, and Generic rule reasoner for supporting user defined rules over RDF graphs

MANU-SQUARE's semantic infrastructure is built on a triple store (GraphDB) that provides a built-in, non Jena-based, RDFS and OWL reasoner. Although with sufficient support for RDFS and OWL reasoning, GraphDB lacks a general-purpose rule reasoner for supporting all possible domain-specific and user defined rules. For that reason, the MANU-SQUARE's semantic infrastructure plan to embed in a Jena Generic rule reasoner, which will allow execution of domain-specific rulesets on named graphs. Jena is more powerful rule language than the rule language of GraphDb, as it has

procedural primitives that can be called by the rules, for example, min, max, sum, product functions, numerical comparison operators, instance creation primitive etc.

Running the Jena rules over a huge triple store, and Manu-Square is expected to generate many triples, can be a computational resource demanding and time consuming operation. The Jena rules support can rather be designed in a way to operate on smaller portions of the triple store, for example on the isolated named graphs. Hence, we plan to provide a database trigger-alike environment for rules execution using Jena engine. In such environment, when a specific action occurs within a semantic infrastructure (e.g. named graph insert or update action), the domain-specific Jena ruleset associated with the action will execute on a single named graph only. For example, there can be a named graph that captures a process chain or supplier description, and before it gets inserted into a semantic store, the ruleset associated with the before-insert or/and after-insert event can be fired on the named graph, and consequentially, can derive new facts for the existing named graph facts.

Jena rule engine provides forward chaining, backward chaining and a hybrid rule execution model, and does so with two internal rule engines; one forward chaining RETE[2] engine and one tabled datalog engine. These model can be run separately (either forward changing mode or backward chaining mode) or the forward engine can be used to prime the backward engine which in turn will be used to answer queries.

We are not planning to implement a support for backward/hybrid rules execution, which is typically used for the query answering tasks, unless there will be very specific requirements and good reasons to do so. Query answering in our infrastructure is supported by a standard SPARQL Query language, over the whole triple store. On another side, we plan to support forward chaining execution of Jena production rules; those rules will be executed on named graphs before or after graph management operations.

A rule in Jena is defined by a Java Rule object with a list of body terms (premises), a list of head terms (conclusions) and an optional name and direction (-> for forward, <- for backward). Each term is either a triple pattern, an extended triple pattern or a call to a built-in primitive. Simplified rule syntax is given in the following listing:

```
Rule      :=   bare-rule .
bare-rule :=   term, ... term -> head term, ... head term    // forward rule
hterm     :=   term
term      :=   (node, node, node)           // triple pattern
          or   (node, node, functor)        // extended triple pattern
          or   builtin(node, ... node)      // invoke procedural primitive
functor   :=   functorName(node, ... node)  // structured literal
```

For example, there can be an inference rule that derives an inverse link (named 'hasProcessChain') of a 'hasSupplier' that links ProcessChain's instance to a Supplier's instance:

```
[invereserPropertyRule: (?P rdf:type core:hasSupplier) (?Y rdf:type core:ProcessChain) (?X
rdf:type core:Supplier)  (?Y ?P ?X)  -> (?X core:hasProcessChain ?Y) ]
```

6.5 MANU-SQUARE domain-specific Inference Rules

6.5.1 Property inference

Some of the properties of instances stored in MANU-SQUARE' semantic repository can be inferred from the asserted statements. Inference is a preferable capablity of manufacturing sourcing platforms such as MANU-SQUARE, as it may

[2] The Rete Match Algorithm is an efficient method for comparing a large collection of patterns to a large collection of objects (facts). It finds all the facts that match each pattern and derives new facts [1]

reduce effort for explicitly populating descriptions of suppliers and their manufacturing capabilities and capacities. Below are few examples of property inference:

Knows property inference

For example, the inference subsystem may infer that one suppliers knows another supplier, by applying an inference rule that specifies the following rule "if supplier A has successfully completed a business process with supplier B, then supplier A knows supplier B, and vice-versa". Obviously, there is no need to assert that supplier 'knows' another supplier, but 'knows' can be automatically inferred by the platform. Even this simple example shows that huge effort to manually maintain 'knows' relationships between suppliers is fully reduced and automatized.

Serves Industry property inference

Manufacturing suppliers' profiles usually provide information about what industries they serve (e.g. Automotive Industry, Office Equipment, Sports Equipment, Aviation Industry, Marine Industry, Kitchen Appliances, Computer Industry, etc). This is a dynamic part of the suppliers profile, and can evolve on a daily basis. With more and more sucesfull business opportunies with customers from different industries , the suppliers' serves-industry description should be expanded. Manual maintaince of servesIndustry information is an additional effort for suppliers on the platform, thus this effort can be reduced with an inference rules states states „if supplier A has done business with customer B, and customer B has stated which industry it belongs to, then supplier A serves industry of customer B“. Also, there can be a rule that infers which industry is served by exploring the properties of the process descriptions (e.g. output products).

6.5.2 Opportunity inference

When describing their processes, manufacturers also define the process input and output. In some cases, output are waste materials that are not considered anymore for other processes. The MANU-SQUARE semantic layers can discover that a waste material from a process of one company can be used as an input for a process of another company. In some cases, the link could be straightforward (i.e. the same material name is used for describing the two processes) but in many cases the link is not explicit and some rules (e.g. created by domain experts/innovators) can be used to infer such a relationship. Once a link is inferred, the system will inform the involved companies that there is an opportunity. This can be presented to the user either as a specific panel of the user interface, collecting emerging opportunities, or as special suggestions during searching/browsing activities of the user.

6.5.3 Capability Selection inference

Selection of appropriate machining capability, process operation, or equipment, if not explicitly stated within an RFQ, can be inferred from a given descriptions and associated inference rules. Similar rules and technique have been also discussed in [2].

For example:

- If an RFQ requires hole aspect ratio (a hole depth-to-diameter ratio) more than 10, then the qualified supplier should provide Deep Hole Drilling process (DeepHoleDrilling capability) for creating the hole. Obviously, a customer does not need to know what process operation is required to produce the hole, but the operation can be inferred using domain-specific rules.
- If an RFQ requires the accuracy of 0.0001 or less, then then the qualified supplier must provide a Precision Service capability.
- If an RFQ requires the tolerance of a hole 0.0001 or less, then hole needs drilling followed by a Reaming operation, thus, the qualified supplier must have a Reaming capability.
- If an RFQ requires reverse engineering, then the qualified supplier must have CAD capabilities.

Such inference rules reveal needed process operations and desired capabilities, and can be easily encoded either with SPIN or Jena, and attached to an RFQ instances.

6.5.4 Machine Classification inference

A classification is a key reasoning technique that is used when selecting the feasible capability or machine tools. Description of machine tools provided by suppliers can have very few statements and can be uncomplete, due a lack of knowledge or simply, due lack of the time to complete descriptions. Thus, the platform can offer inference rules to classify an equipment, based on its features, attributes, and KPIs.

For example:

- If an instance of machine tool has xAxis, yAxis, and zAxis attributes, is controlled by computer, then a machine tool is classified into 3AxisCNCMachine.
- If a supplier provides cut sheet metals operation with thickness larger than 1 inch, the the supplier must have a water jet cutting machine / capabilities

In addition to equipment classification inference, the platform can offer to automatically complete descriptions of instances of certain equipment type, based on already established complete description of instances of the equal equipment type from other suppliers. However, this cannot be done without explicit permission of suppliers to share and reuse their descriptions.

6.5.5 Inference to reduce unnecessary explicit classifications

We plan to set up inference rules to derive facts that do not necessarily have to be asserted for instances of MANU-SQUARE core concepts, but that rather can be inferred using the semantics of the MANU-SQUARE core concepts and their relationships. For example,

- There is no need to explicitly asset that a resource type is an Assembly. Simply, an inference rule can define that if resources X and Y are Items, and X is linked to Y via hasItem property, then infer that X is an Assembly.
- There is no need to explicitly asset that a resource type is a Component. Simply, an inference rule can define that if resource X is an Item, and X has no hasItem property, then infer that X is a Component.
- Further, if X is an Item, and X is linked to a process with hasInput relationship, then infer that X is an InputItem. Or, if X is an Item, and X is linked to a process with hasOutput relationship, then infer that X is an InputItem
- If something is a Material, and is directly used as an input or output for a process, then infer that material is also an Item.

These rules can be implemented in different ways, either by using SPARQL CONSTRUCT, SPIN rule syntax, Jena rule or GraphDB Custom rule. We have tried already to implement some of them with custom rules added into RDFS ruleset of GraphDB and it proved working sufficiently fine, in real-time. Below are examples of these custom rules added into RDFS ruleset of GraphDb:

```
Id: inputItem classification
  x <rdf:type> core:Process
  x <core:hasInput> y
 --------------------------------
  y <rdf:type> core:InputItem

Id: outputItem classification
  x <rdf:type> core:Process
  x <core:hasOutput> y
 --------------------------------
  y <rdf:type> core:OutputItem


Id: Classification of material into items#1
  x <rdf:type> core:Process
  x <core:hasInput> y
  y <rdf:type> core:Material
 --------------------------------
  y <rdf:type> core:Item
```

```
Id: Classification of material into items#2
  x <rdf:type> core:Process
  x <core:hasOutput> y
  y <rdf:type> core:Material
 -------------------------------
  y <rdf:type> core:Item
```

7 GITLAB

During MANU-SQUARE project, consortium decided to use GITLAB tool to upload all technical activities documents.

Accessing to https://gitlab.com/gitlab-org it's possible to register and then get access to all documentation.

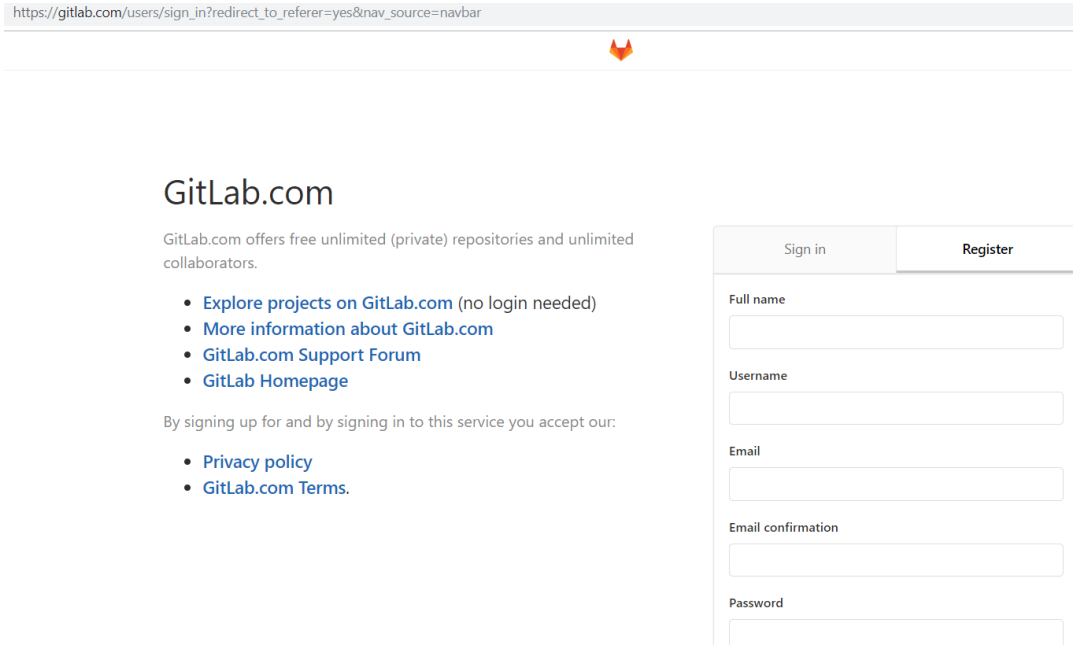

Figure 21: GITLAB register homepage

Once a user register himself, he can send an email to research@holonix.it asking to be part of "MANU-SQUARE project, Group ID: 5374823".

Holonix will give access and then it will be possible to view all uploaded documents.

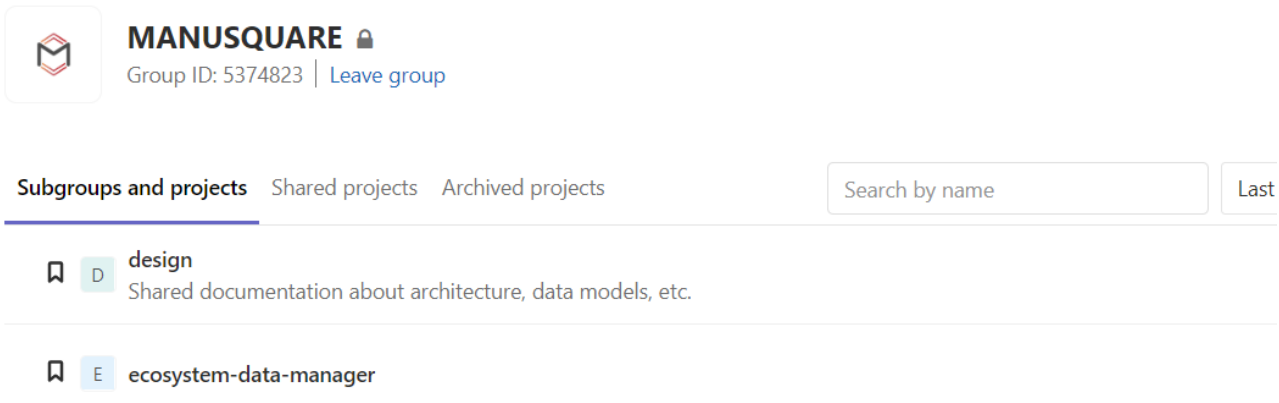

Figure 22: GITLAB MANU-SQUARE project home page

Clicking on "Design" it's possible to have access to all related documents as Data Models, Inference, etc.

Figure 23: Data models home page in GITLAB

8 CONCLUSION

Main goal of this deliverable has been the definition of the extended MANU-SQUARE platform data model. Starting from the work done, it's possible to develop the MANU-SQUARE platform with a shared and consistence base.

In this document the work done by all partners, involved in tools' development, has been highlight, the extended data model, per each specific tool, is reported, as well as the final version.

It has to be considered that the content of this deliverable could be updated and slightly changed during the development of tool. Updates regarding data models will be inserted in deliverable of WP4, regarding tools architecture and implementation. Similarly, updates regarding inference rules will be inserted in deliverable D2.4.

9 REFERENCES

[1] Forgy, Charles L. "Rete: A fast algorithm for the many pattern/many object pattern match problem." Readings in Artificial Intelligence and Databases. Morgan Kaufmann, 1989. 547-559.

[2] Sadeghi, Samira, and Farhad Ameri. "An Intelligent Process Planning System Based on Formal Manufacturing Capability Models." ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2013.