MONSOON

MOdel based coNtrol framework for Site-wide OptimizatiON of data-intensive processes

# D2.7–Initial Cross-sectorial Domain Model

| | |
|---|---|
| Deliverable ID | **D2.7** |
| Deliverable Title | **Initial Cross-Sectorial Domain Model** |
| Work Package | **WP2 – Requirements Engineering and Reference Architecture** |
| Dissemination Level | **PUBLIC** |
| Version | **1.0** |
| Date | **2017-07-26** |
| Status | **final** |
| Lead Editor | |
| Main Contributors | **Martin Sarnovsky, Peter Bednar (TUK), Massimo de Pierri (LCEN), Christian Beecks (FIT)** |

**Published by the MONSOON Consortium**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723650.

## Document History

| Version | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 2017-07-26 | Martin Sarnovsky(TUK) | First Draft with TOC. |
| 0.2 | 2017-09-02 | Martin Sarnovsky (TUK) | Overview of existing standards for data analytics and production process modelling. |
| 0.3 | 2017-09-12 | Peter Bednar (TUK) | Initial description of cross-sectorial model. |
| 0.4 | 2017-09-13 | Peter Bednar (TUK) | Evaluation chapter. |
| 0.5 | 2017-09-18 | Peter Bednar (TUK) | Added description of the classes and properties. |
| 0.6 | 2017-09-25 | Peter Bednar (TUK), Massimo De Pierri (LCEN) | Added Motivation and main objectives and Standards for Life-Cycle modelling chapters. |
| 0.7 | 2017-09-27 | Martin Sarnovsky (TUK) | Added examples, B2MML process description, references. |
| 0.8 | 2017-09-29 | Christian Beecks (FIT) | Added SenML and OGC SensorThings API for Data Element modelling. |
| 1.0 | 2017-10-05 | Peter Bednar (TUK) | Final integration and corrections. |

## Internal Review History

| Version | Review Date | Reviewed by | Summary of comments |
|---|---|---|---|
| 0.8 | 2017-10-03 | Marco Dias [GLN] | Fully accepted |
| 0.8 | 2017-10-05 | V. Bonnivard [PROB] | Minor typos / layout corrections |

## Table of Contents

# 1 Introduction

Main objective of the work presented in this deliverable is to define the initial semantically-enriched model that serves as a basis upon which all other models will be developed. Since the MONSOON platform will integrate various systems and technologies, proper representation of the information and data is required. Semantic model will cover production processes, which will be modelled using functional blocks with specified inputs and outputs. Model will cover the data pre-processing steps as well as predictive functions. The semantic model will leverage existing standards for description of both production processes and predictive functions.

The deliverable is organized as follows:

First, we provide a brief overview of the existing relevant technologies used for semantic modelling. Next chapter describes existing standards in area of manufacturing processes, predictive functions and standards related to modelling of life-cycle engineering. Then, the structure of Initial cross-sectorial domain model is presented. We provide the overall model structure description as well as description of the particularconcepts and their respective properties.

## 1.1 Scope

This document presents the Initial cross-sectorial domain model used in MONSOON platform. This document is a part of the *"Task 2.5 – Cross-sectorial domain model"* which is meant to develop the model definition. The main objective is to develop a model, that will unify existing standards for data analysis (e.g. PMML) with standards related to manufacturing processes and their linking with ERPs (e.g. B2MML). Model will be supported and maintained by Semantic framework for dynamic multi-scale industry modelling developed within Task 4.1, which will provide the tools to model the concrete production processes, etc. Initial model developed in deliverable *"D2.7 - Initial Cross-sectorial Domain Model"* will be extended and refined into the final model that will be presented in deliverable "*D2.8 – Final Cross-sectorial Domain Model"*.

## 1.2 Terminology remark

The term **model** and **modelling** in this document is used in the following different meanings:

a) as the **semantic model**, i.e. formal specification of the concepts from the given domain
b) as the **data analysis model**, i.e. result of applying of the data modelling techniques (e.g. predictive function).
c) as the **process model**, i.e. model which specifies decomposition of the process to steps and describes their data and execution dependencies (usually based on graphical notation in the form of flow charts).

# 2 Motivation and main objectives

This introductory chapter describes the main objectives of the Cross-sectorial domain model in the context of the general data analysis process.

## 2.1 Process of data analysis

The process of data analytics can be described by the standard methodologies such as CRISP-DM or SEMMA [1, 2], which breaks the process of data analysis into six major steps:

- **Problem understanding**–This initial phase focuses on understanding the objectives and requirements of the data analysis from a business perspective, i.e. how the production process can be optimized by data analytics methods.During this phase, the domain experts in cooperation with the data scientists analyse overall production process and its steps (production segments) and specify which key-performance indicators will be optimized. This knowledge is then converted into the data analysis problem definition (e.g. classification, regression, anomaly detection etc.) by data scientists.

- **Data understanding**–The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.During this phase, data scientists with the cooperation of domain experts will select and describe the subset of relevant data required to achieve optimization objectives specified in the phase of Problem understanding. The part of data description is also identification of the already known dependencies between the data and key-performance indicators.

- **Data preparation**–The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modelling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modelling tools.In the MONSOON project, all data are collected and processed in the Data Lab platform. In order to collect initial raw data from the production site environment, data scientists have to cooperate with the site IT-specialists responsible for the setup of the data integration components.

- **Modelling**–In this phase, various data modelling techniques (such as decision trees, neural networks, logistic or linear regression models etc.) are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data analysis problem type. Some techniques have specific requirements on the form of data. Therefore, stepping back to the Data preparation phase is often needed. Inthe MONSOONproject, data modelling is performed on the Data Lab platform. The result of this phase is the set of data analytics models (i.e. predictive functions) that appears to have high quality from a data analysis perspective. This quality is usually evaluated by applying the model to the independent testing data set not used during the building of the model and computing the evaluation statistics such as classification accuracy (number of tested records with the correctlypredicted value by the model/ total number of records in the testing dataset) etc.

- **Evaluation**– Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. Within this phase, quality of the models evaluated on the testing

dataset is projected into the business oriented key-performance indicators specified in the Problem understanding phase. For example, during the Problem understanding phase, domain experts estimated that if some defect in the product can be detected beforehand during the production process it is possible to save specified amount of waste materials. During the Modelling phase, data scientists build a predictive function and estimated its classification accuracy to 98%. During the Evaluation phase, the models (predictive functions) are evaluated by computing the impact of predictive model to the specified key-performance indicators from the evaluation statistics.At the end of this phase, a decision on the use of the data analysis results should be reached.

- **Deployment**–During the Deployment phase, predictive function is packed, exported from the Data Lab platform and deployed in the production site environment. Important part of the Deployment phase is also specification of the monitoring and maintenance of the model in the production conditions.

The sequence of the phases is not strict and moving back and forth between different phases is always required. Note that the process of data analysis is generic and can be used in different domains (i.e. the same phases have to be implemented for plastic domain or aluminium domain).

## 2.2 Main objectives for Cross-sectorial domain model

The main objectives of the Cross-sectorial domain model are to provide the tools to:

- Improve communication between the data scientists and domain experts in phases of Problem understanding, Data understanding and Evaluation by creation of the shared knowledge base.
- Allow within and cross-sectorial sharing of knowledge about the optimization of the production processes by data analytics methods and predictive functions.
- Automatize Evaluation of predictive functions by automatic computation of impact on the key-performance indicators from evaluation statistics.
- Allow "what-if" simulation scenarios and automatic optimization of production process during the Modelling and Evaluation phases.
- Automatize validation of data dependencies during the Deployment of the predictive function in the operation environment.

In order to achieve these objectives, Cross-sectorial domain model support:

a) Presentation in the human-readable form using the Semantic framework tools developed in work package 4. Semantic framework tools allow creation, visualization, querying, filtering and navigation of the Cross-sectorial domain model elements by data scientists and domain experts.
b) Machine-readable format with the formally defined semantics, which support automatic validation of deployment, quality evaluation of the predictive functions and simulation and automatic optimization of the production processes.

The main part of this document describes formal specification of the Cross-sectorial domain model and its machine-readable format. Human-readable representation and graphical notation for modelling will be specified in deliverable D4.1. The Cross-sectorial domain model is formally specified as the Resource Description Framework (RDF) vocabulary (see the following chapter about the overview of technologies for semantic modelling). Machine-readable format is specified as the JSON-LD schema, which combines JSON data format with the semantic RDF annotations.

Note that Cross-sectorial domain model do not specifies concepts for the specific domain, e.g. specific data elements or KPIs for plastic industry, etc. Cross-sectorial domain model is a meta-model, which specifies meta classes for the formal description of these specific concepts (i.e. it specifies for example "type" DataElement for description of all specific data elements).

The domain model is formally divided to the three modules:

- Module for Production process modelling.
- Module for Data and KPI modelling.
- Module for modelling of the Predictive functions.

Data and Predictive function modules are further divided to "logical view" and "physical view". In logical view, data elements and predictive functions are specified on the conceptual level without any details how the data required to build a predictive function have to be integrated in the site environment and collected, stored and pre-processed in Data Lab platform. Logical view also serves for sharing of the knowledge and transferring the predictive functions to new site environment.

The following table summarizes which modules are relevant to the particular phase of the data analysis process.

| Data analysis phase | Cross-sectorial domain modules | Involved roles |
|---|---|---|
| Problem understanding | Production process modelling, logical Data and KPI modelling, logical Predictive function modelling | Domain experts, Data scientists |
| Data understanding | Physical Data modelling | Domain experts, Data scientists, IT experts |
| Modelling | Physical Predictive function modelling | Data scientists |
| Evaluation | Physical and logical Data modelling and Predictive Function modelling | Domain experts, data scientists |
| Deployment | Physical Data modelling and Predictive function modelling | IT experts |

**Table 1 – Mapping between the Cross-sectorial domain model and phases of data analysis process**

# 3 Existing technologies for semantic modelling

Semantic technologies usually refer to a set of World Wide Web Consortium (W3C) standards aimed to support of exchange of information about the data and relationship in data in a format which is readable by machines. Semantic technologies usually cover data models, querying tools, systems for data management, etc. Following sectionsdo not try to give a complete survey of existing technologies, but rather introduce the main ones, that may be relevant to the MONSOON project. We briefly introduce the RDF (and RDF Schema) as a standard which is often serialized using XML and JSON-LD used for serialization of Linked Data in JSON format.

## 3.1 RDF and RDF Schema

The RDF (Resource Description Framework) is a standard and general method for simple conceptual descriptions of information publishedin web resources [3]. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web. What XML is for syntax, RDF is for semantics - a clear set of rules for providing simple descriptive information. RDF is a W3C recommendation since 1999.

The RDF data model is based on modelling of the statements about the resources. The statements are in the form of subject-predicate-object expressions, sometimes called as the triples. Resource is represented by the subject and its particular aspect or property is represented by the predicate, which specifies the relation between the subject and the object (e.g. value of the property). For example, a simple statement “Aluminium is produced from alumina" can be represented as the RDF triple with "Aluminium" as the subject, “produced from"as the predicate and "alumina" as the object. The predicate is represented by URI representing the relationship and the object can be represented by a URI or a literal. RDF statements can be represented in a form of graph.

Several serialization formats are available, e.g. turtles, n-triples, JSON-based formats and RDF/XML. The subject is a unique element, represented by its URI (Uniform Resource Indentifier). RDF statements can be queried using RDF query languages. There are multiple options available e.g. SPARQL, RDQL, etc.

RDF strength is in its descriptive capabilities, but is still lacks some important features required for ontology modelling.RDF does not provide any mechanism for declaration of the concepts such as classes, properties and relations. The main goal of the RDF-Schema (RDF-S) was to extend the RDF and to provide the capabilities to construct the ontologies [4]. Main RDF-S concepts include classes and associated properties. Rdfs:Class declares a resource as a class for other resources. Properties represent the relation between the subject resources and object resources.

RDF Schema then provides a way for those descriptions to be combined into a single vocabulary. RDF enforces a strict notation for the representation of information, based on resources and relations between them. Properties describe the relations between the resources (between the subject and object) and basically represent the predicate. Properties can be specified by their range and domain (rdfs:range and rdfs:domain properties), which describes the types of applicable classes. rdfs:type property can be used to state, that the resource is an instance of particular class and rdfs:subclassOf allows to create hierarchical structures of the classes (in similar fashion, rdfs:subPropertyOf can be used to properties).

## 3.2 JSON-LD

JavaScript Object Notation for Linked Data (JSON-LD) is a lightweight format for serialization of Linked Data using JSONformat [5].The RDF can be sometimes hard to read and write and can cause overhead when using in web applications, while JSON is widely used but lacks some functionalities and features available in RDF (linking between different data sources, ontologies, etc.). Motivation was to provide the format which unifies

RDF and JSON by enabling the expressions supported by RDF and providing JSON syntax. One of the main goals wasto minimize the effort as possible from developers when transform from existing JSON to JSON-LD - JSON-LD is 100% compatible with JSON and therefore numerous parsers and libraries can be reused. JSON-LD is a W3C recommendation for representation of the RDF data as JSON objects since 2010.

JSON-LD is from the structure point of view same as other JSON documents. JSON-LD extends the traditional JSON with specific reserved names using prefix "@", such as *@context* or *@type*. The concept of a context is used to provide additional mappings from terms in JSON to IRI (Internationalized Resource Identifier). In other words, the context is used to link the object properties in a JSON document to concepts in a referenced vocabulary or ontology. The context can be embedded directly into a JSON-LD document or can be referenced from an external separated file.Type specification can be used to specify the type of the item. JSON-LD enables to annotate the information about the item type using Item Properties and Values. Item property is used from the specified vocabulary/ontology (and must belong to properties allowed within the item type). Value then represents the particular property value (or multiple values). JSON-LD also supports nested properties.

JSON-LD allows developers to focus on JSON and modellers to focus on RDF data model. Using of JSON-LD does not require any knowledge of RDF, but can be used that way, e.g. when using in combination with SPARQL or is integrated with similar technologies. JSON-LD can be used to serialize any RDF graph and most of JSON-LD documents can be described in RDF.

# 4 Overview of existing standards

This section will provide an overview and description of already existing standards relevant to MONSOON Cross-sectorial domain model which will be introduced later. Some of those relevant standards (or their parts) can be integrated to the cross-sectorial domain model. We decided to focus mostly on the most important areas - modelling of manufacturing processes, modelling of predictive functions (and data pre-processing) and standards covering life-cycle aspects. Following sub-sections will give an overview of most commonly used modelling standards in those domains.

## 4.1 Standards for modelling of manufacturing processes

### 4.1.1 B2MML

Business To Manufacturing Markup Language (B2MML) is an XML-based set of schemas for modeling of the manufacturing processes [6]. It was published by Manufacturing Enterprise Solutions Association (MESA) and it is a completeXML implementation of the data models in ANSI/ISA-95 standard, which is international standard for developing an automated interface between enterprise and manufacturing control systems. ISA 95 standard was developed to be applied in all industry domains and in all sorts of processes. B2MML is frequently used to integrate business systems (e.g. ERP systems) with manufacturing systems (e.g. control systems). Currently, V0600 is the latest release of the B2MML standard.

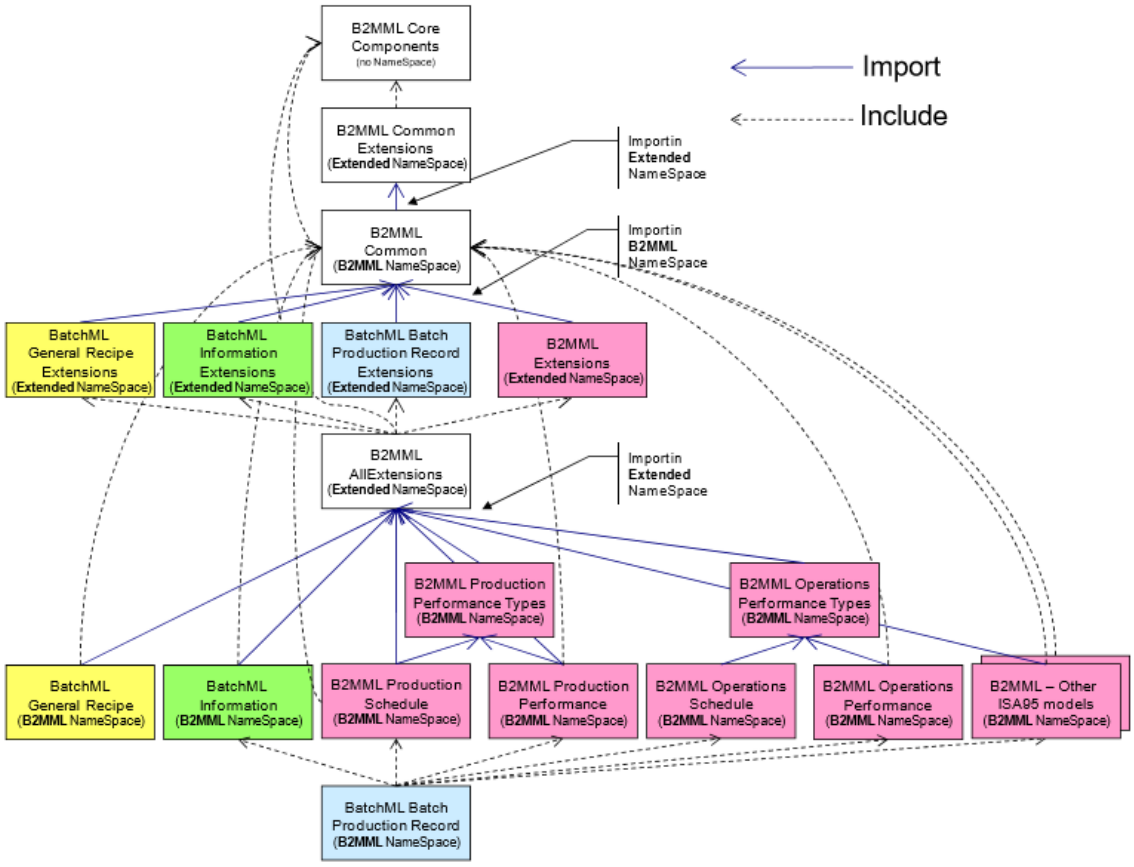


**Figure 1 – Overall structure of the B2MML model [6].**

General schema of the B2MML model is depicted on Fig. 1. Common ISA88 and ISA 95 schemas are represented by the white boxes. Pink boxes refer to the B2MML specific schemas representing the ISA 95

standard. Blue, green and yellow boxes represent the BatchML schemas. Next sections will describe the models relevant to production processes.

In general, B2MML allows to describe the following groups of information related to manufacturing processes:

- Processes - Process segments.
- Resources - Material, Personnel, Equipment, Physical Assets.
- Production - Definition, Capability, Schedule, Performance.
- Operations - Definition, Capability, Schedule, Performance.
- Work - Definition, Capability, Schedule, Performance.

### Process Segments

*Process segment* is a logical grouping of personnel resources, equipment resources, and material required to carry out production step. Process segment also may define specific resources, such as specific equipment needed and resources quantity. Process segments are identified by analysis of overall manufacturing process and business processes implemented in the organization.
In general, three types of process segment are available:

- Production segments – segments related to conversion of materials to final products (or intermediate materials).
- Movement segments – segments related transportation of materials and tracking of their location.
- Inspection segments – segments related to materials and products quality and quality testing.

### Resources

*B2MML Material*
Consists of*MaterialLot*specifying material, countable or weighable, which describes the actual total quantity or amount of material available, its current state, and its specific property values. *MaterialDefinition* is a means to describe goods with similar characteristics for purposes of scheduling and planning, e.g.“Grade B Aluminum”. The materials may be identified as raw, intermediate, or final and may have other state information(e.g. availability of safety information). *MaterialClass* is a means of defining groupings material definitions for use in production scheduling or processing e.g. "Sweetener", with members of "Fructose"and "Sugar Cane Syrup".*MaterialTestSpecification* may be associated with a material class property if a test is required to ensure that the material has the required property value.

*B2MML Personnel*
*Person* represents a specifically identified individual involved in production process and contains information about specific personnel.*Personnel Class*describes a group of persons with similar characteristics. Any person may be a member of zero or more personnel classes. Examples of personnel classes can be “Slicing Machine Operators” or “Zipper Line Inspectors”. *Qualification test specification* may be associated with a personnel class property to describe if a qualification test is required for specific operations.

*B2MML Equipment*
*Equipment* represents the elements of the equipment hierarchy model defined in ANSI/ISA-95.00.01.Equipment concept cover sites, areas, production units, production lines, work cells, process cells, or units. Equipment may consist of other equipment, e.g. production line may be made up of work cells. Each may be defined as a separate equipment element with separate properties and capabilities.*Equipment class*describes a group of equipment with similar characteristics. Similar to Personnel class, *Equipment capability test specification* may be associated with an equipment to ensure that the equipment has the rated capability.

*B2MML Process Segment*

*Process segment* is a logical grouping of personnel resources, equipment resources, and materials required to carry out a production step. It defines what types of personnel, equipment, and material are needed and it can describe specific resources (e.g. some specific type of equipment for particular segment) and quantity of needed resources. Identifying process segments requires an understanding of the business processes within the company and the general structure of the manufacturing processes. Three general types of process segments are supported:

- *Production segments* – segments related to conversion of raw materials into intermediate materials or final products.
- *Movement segments* – segments related to movement of materials and keeping track of material and product locations.
- *Inspection segments* – segments related to confirming or testing quality and suitability of materials and products.

**Production**

*Product Definition:*

Describes how to produce a product. *ManufacturingBill*specifiesthe material in the manufacturing bill - quantity of the material needed, an identification of the material class, any manufacturing bill item assemblies, and the corresponding bill of material ID, etc.*ProductSegment*defines a product segment, including resources required for the segment, an estimated duration of the segment, an identification of the corresponding process segment, parameters associated with the segment, the segment dependencies, and any encapsulated segments.*EquipmentSpecification*defineswhat equipment resources required for the product segment and*MaterialSpecification*specifies the material resources required (as well as *Personnel* and *PhysicalAsset*).

*Production Capability*

Provides information about all resources needed for production for selected times and within a selected site, area, process cell, production unit, or production line. It consists of *PersonnelCapability*, *EquipmentCapability, PhysicalAssetCapability*, *MaterialCapability*, they are defined as a set of references to material, personnel, equipment and assets capacities that are unused/committed, available/unavailable.

*Production Schedule*

The production schedule contains the information that defines the context of the schedule, such as start time, end time, location, and published date. Consist of *ProductionRequest*, a request for production for a single product identified by a production rule. A production request contains the information required by manufacturing to fulfill scheduled production.

*Production Performance*

The production performance contains the information that defines the context of the report, such as start time, end time, location, and published date. *ProductionResponse*is the response from manufacturing that is associated with a production *Request*. A production result may include the status of the request, such as the percentage complete, a finished status, or an aborted status. The production response for a specific segment of production is defined as a *SegmentResponse*. *ProductionData* contains a definition of a production data element, corresponding to a process segment or product segment parameter. *PersonnelActual*in a production response identifies a personnel resource by used during the specified segment of production. *EquipmentActual*in a production response identifies an equipment resource used during the specified segment of production. *PhysicalAssetActual*in a production response identifies a physical asset resource. Material produced, material consumed, or consumable materials are identified in a *MaterialActual*.

**Operations**

*Operations Definition*

*OperationsSegment* information defines what personnel, equipment, physical asset, or material resources are required for execution of the operations segment. It does this by defining the classes of resources, or in some cases the exact instance of a resource required, e.g. an assembly segment may require 1 assembler for 2 hours, and 1 assembly machine for 2 hours. In some industries, the exact assembly machine may have to be specified, such as "AssemblyMachine#1". Consists of *EquipmentSpecification* - definition of the equipment resources required for the product segment, *MaterialSpecification* as the definition of the material resources required for the product segment, as well as *Personnel* and *PhysicalAsset* specification.

*Operations Capabilities*

*Operations capability* is the collection of information about all resources for production for selected times and within a selected site, area, process cell, production unit, or production line. This is made up of capability information about *equipment*, *physical assets*, *material*, *personnel*, and *processsegments*. Operations capability also defines the available capability, committed capability, and unattainable capability of each resource, and each resource within a process segment.

*Operations Schedule*

*Operations schedule* is made up of a set of 1 or more *Operations requests*. The Operations schedule also contains the information that defines the context of the schedule, such as start time, end time, location, and published date. *Operations request* defines a request for *Operations*. An operations request identifies the associated *Work Definition*.

*Operations performance*

*Operations Performance* report is made up of a set of 1 or more operation responses. *Operation responses* are the response from operations that is associated with an *Operations Request*. There may be one or more operation responses for a single operation request if the facility needs to split the request into smaller elements of work. A result may include the status of the request, such as the percentage complete, a finished status, or an aborted status.

**Work**

*WorkMaster* and *WorkDirectives* elements define the exchange information structure for a *Work Master and Directives*, as defined in ANSI/ISA95 Part 4. Work Performance, Schedule and Capability schemas in similar fashion as corresponding schemas for Operation and Production.

## 4.2 Standards for modelling of predictive functions

In this section, we will briefly describe the most commonly used standards for modelling of the predictive models and functions. We will describe PMML as an XML-based standard and PFA as a JSON based one.

### 4.2.1 PMML

PMML (Predictive Model Markup Language) is an XML-based predictive model interchange format [7]. PMML provides a way to describe the predictive models (e.g. produced by machine learning algorithms) in order to enable the sharing of the models between the tools and applications or between different environments (e.g. from development to production environments). It supports most common predictive models as well as various data transformations and verification methods.PMML was developed by Data Mining Group (DMG) and its latest version is 4.3, released in August 2016.PMML is XML-based, the structure of the document consists of pre-defined elements and attributes which represent the structure of the predictive task - from data processing and transformations to the predictive models. PMML supports most frequently used models (see details in following sections) and data transformations and pre-processing methods. It also supports several built-in functions that allows data manipulation. Following paragraphs briefly describe the main components of PMML.

*Header*
This component contains general information about the PMML (e.g. description, information about the application that was used to create the model, name, version, timestamp, etc.).

*Data Dictionary*
Contains definitions of fields used in the model (categorical, continuous, ordinal) and their ranges. *DataDictionary*is a top-level tag and contains *numberOfFields* which specifies number of fields in the dictionary. *DafaField* specifies characteristics ofparticular field, name and type. It re-uses the atomic types in XML Schema, introduces a few new ones (date and time related), Data Dictionary also specifies the type of operation available on the data, and supported values and ranges.

*Data Transformations*
Describes the data pre-processing, e.g. normalization, discretization, value mapping, custom and built-in functions, aggregation, etc. PMML allows the definition of the Constant values and supports those data transformations:

- Normalization: transform values to numeric values.
- Discretization: transform continuous values to discrete values.
- Value mapping: mapping of discrete values to discrete values.
- Text Indexing: derive a frequency-based value for a given term.
- Functions: derive a value by applying a function to one or more parameters.
- Aggregation: summarize or collect groups of values (compute average).
- Lag: use a previous value of the given input field.

*Statistics*
Contains the schema for description of several statistic methods. The statistics for a model can consist of the statistics for a particular fields or statistics for fields in the presence of other fields (e.g. ANOVA method, computation of correlations between fields, etc.).

*Taxonomy*
Enables the description of the categorical fields organized in hierarchies. The representation of hierarchies in PMML is based on parent/child relationships. A tabular format is used to provide the data for these relationships.

*Mining Schema*
*MiningSchema*is a list of fields used in the model.It contains specific information about particular field and information specific to a certain model (*DataDictionary* contains data definitions which do not vary per model). Main purpose is to list the fields that have to be provided in order to apply the model (e.g. if it is active as the input variable, target, predicted, etc).*MiningSchema* also specifies the outlier treatment, handling of the missing values.

*Model*
Model contains definition of the model itself, based on the type of the model contains various attributes describing the particular model. PMML allows description of most commonly used predictive models including SVM models, association rules, Naïve Bayes classifiers, Tree Models, Regression models and Clustering models. The list of the supported models in the PMML 4.3. version:

- Association Rules.
- Baseline Models.
- Bayesian Network.
- Cluster Models.

- Gaussian Process.
- General Regression.
- k-Nearest Neighbours.
- Naive Bayes.
- Neural Network.
- Regression.
- Ruleset.
- Scorecard.
- Sequences.
- Text Models.
- Time Series.
- Trees.
- Vector Machine.

*Targets*
This component describes the targets of the predictive model. The target values are derived from the particular elements in the models (e.g. the target categories in RegressionModel are specified in the RegressionTable elements, the TreeModelare definedin the Node elements).

*Output*
This component is used to describe the output fields (particular results that can be returned from the model)such as predicted class in classification or many other e.g. probability, standard error, entity ID, support, lift, confidence, ruleValue, clusterID, ClusterAffinity etc. Each model has specified a set of supported outputs.

*Model Verification*
*ModelVerification* provides a mechanism to ensure that if the model is deployed into a new environment, it will generate results consistent with the environment where the model was created. The producer of a PMML model has to add a set of verification records to the model to support the verification, e.g. meaningful sample of the training dataset, including normal cases as well as exception cases, such as missing data, outliers. etc.

*Model Explanation*
The elements of this component can be used to evaluate/explain the quality of the model depending on the type of the model. In PMML 4.3. supports most commonly used evaluation techniques:

- Univariate Statistics.
- Partitions.
- Predictive Model Quality.
- Clustering Model Quality.
- Gains/Lift Charts and Ranking Quality Information.
- ROC (Receiver Operating Characteristic) Graph.
- Confusion Matrix.
- Field Correlation.

### 4.2.2 PFA

Portable Format for Analytics (PFA) is an JSON-based predictive model interchange format [8]. PFA enables to describe and exchange predictive models produced by machine learning algorithms. PFA in general helps with the transition of the predictive models from development environment into the production. Supported development environments can produce the models as a JSON documents with a structure defined by PFA (e.g. machine learning algorithm from developer environment produces a classifier trained on training data in

PFA format, that can be executed in the production site on live data).Fig. 2 depicts the transition from the development environment into the production one and includes various supported tools. PFA was developed by the Data Mining Group (DMG) and is complimentary to the DMG's XML-based PMML standard.Similar to PMML, PFA is also an intermediate text file to be deployed in the production environment. Main difference is, that the PFA is more flexible than PMML, as the PFA supports creation of new types of models from building blocks of the PFA without a need to wait for the new models to be added to the specification.

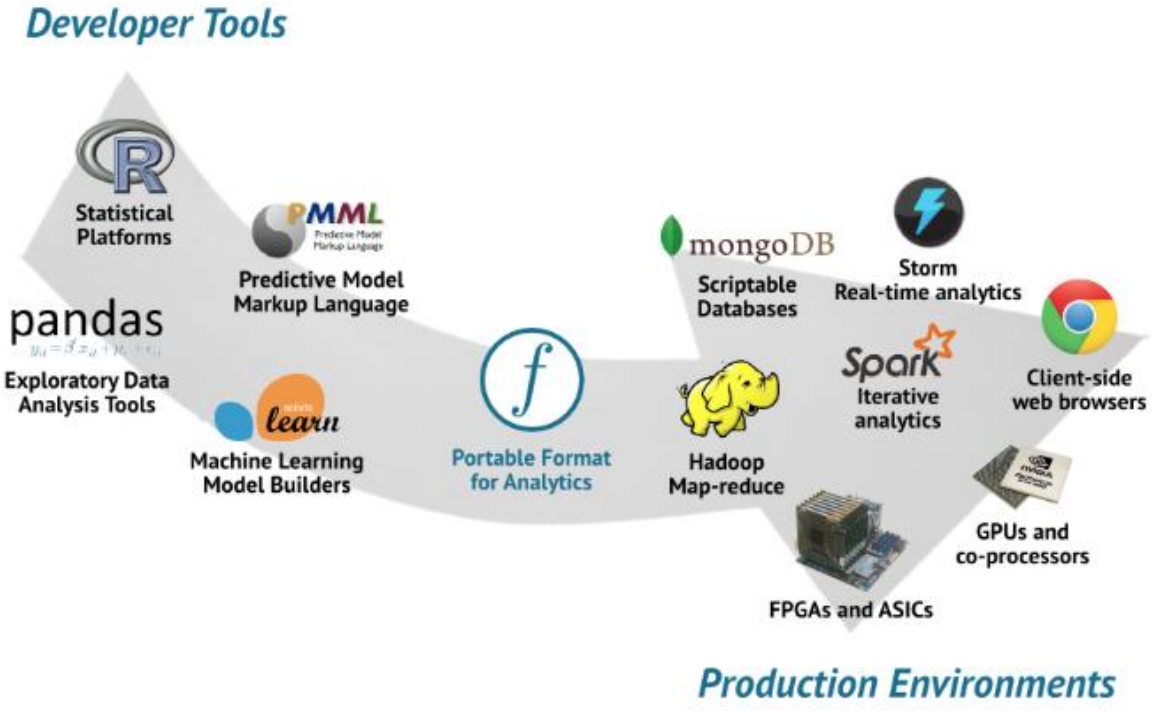


**Figure 2 – PFA supported technologies.**

PFA uses JSON serialization and the PFA JSON object represents an executable scoring engine. PFA supports the control structures including loops, conditionals as well as user-defined functions. Each PFA document has to contain three top level fields required for every PFA implementations: *action*, *input* and *output*. The rest of the fields are optional, non-mandatory. Following section will briefly describe most of the PFA supported fields.

### Required fields

*input*
Specifies input data - contains an Avro schema that represents the data type of data provided to the scoring engine.
*output*
Specifies output of the model - described by Avro schema that represents the data type of data produced by the scoring engine. The way that output is returned to the host system depends on the method.
*action*
An expression or array of expressions that are executed for each record of input data when the scoring engine runs.

### Optional fields

*name*
Optional name used to describe and identify the scoring engine
*method*

Method used to input data. PFA supports map, emit, and fold methods. Map is basically a mathematical function - performs a function on the input data and present it as the output. Emit methods are used to build filters or are used for generation of the tables and fold is used for aggregations. Ifabsent, the default value is map.

*begin*

Contains an expression or array of expressions that are executed in the begin phase of the scoring engine's run.

*end*

Contains an expression or array of expressions that are executed in the end phase of the scoring engine's run.

*fcns*

Contains a JSON object whose member values are function definitions, defining routines that may be called by expressions in begin, action, end, or by expressions in other functions.

*zero*

Embedded JSON data whose type must match the output type of the engine. This must be present in the "fold" method initialize the fold aggregation, and it must not be present in the "map" or "emit" method.

*merge*

An expression or JSON array of expressions that may be executed if the scoring engine container needs to combine partial results from independent "fold" engines. It must be present with the"fold"method and it must not be present with the "map" or "emit" method.

*cells*

A JSON object whose member values specify statically allocated, named, typed units of persistent state or embedded data.

*pools*

A JSON object whose member values specify dynamically allocated namespaces of typed persistent state.

*randseed*

An integer which, if present, sets the seed for pseudorandom number generation.

*doc*, *version*, *metadata*

The optional parameters - used to describe the scoring engine or version.

PFA uses the same type system as the Avro format. Avro data types are described using JSON objects and PFA includes Avro as a language subset. The type system supports:

- Primitives:
  - Boolean - not a subclass of integers, two values, true and false.
  - Integer - two integer types, int and long for 64-bit integers.
  - Floating-point - also two types, float and double.
  - String - accepts any valid Unicode sequence.
- Null type - e.g. used to represent missing data.
- Arrays - ordered collection of items.
- Maps - unordered key-value pairs, keys are string type.
- Records - collection of a fixed set of fields.
- Enumeration sets - small, finite set of strings.
- Byte sequences - bytes, accept any byte sequence.
- Fixed-width byte sequences - named raw byte sequences with fixed width.
- Tagged unions.

### 4.3 Standards for Life-Cycle modelling

There are no existing standards specifically defining environmental KPIs. Life cycle scientist usually refer to technical background documentation provided by international councils; IPCC and JRC[1] are the most active organizations dealing with definition of impact indicators. In particular, ELCD2 database provided by JRC can be the main reference for MONSOON, due to its EU-centered modelling approach.

The ultimate set of potential environmental impact indicators will be defined within WP7 activities and its validation will be one of the final actions of the project. The following table provides an overview of robust indicators that can be used to evaluate the environmental performance in the two industrial domains investigated by MONSOON:

| Indicator | Scale | Acronym | Unit |
|---|---|---|---|
| Global Warming Potential | Global | GWP | $CO_2$ equivalent, kg |
| Acidification Potential | Local | AP | $SO_2$ equivalent, kg |
| Net use of fresh water | Local | FW | $m^3$ |
| Cumulative Energy Demand | Global | CED | MJ |
| Cumulative Waste Production | Local | CWP | kg |

**Table 2– Example of environmental indicators.**

[1] More information available at http://www.ipcc.ch and https://ec.europa.eu/jrc/en
[2] Detailed information about ELCD is available at http://eplca.jrc.ec.europa.eu/ELCD3/

# 5 Initial Cross-sectorial domain model

This chapter provides initial formal specification of the Cross-sectorial domain model. The specification is divided to the following sub-modules:

- Production process modelling – specifies decomposition of production processes to segments (production phases) and describes resources required for the production.
- Data modelling – specifies concepts for description of the data elements and key-performance indicators.
- Predictive function modelling – specified concept for description of the data analytics models.

The description of each module is broken down into thefollowing sections:

- Preamble description with the informal introduction of the module.
- Graphical UML notation for main concepts and relations.
- Vocabulary of URIs specified for the module.
- Formal specification of module classes and properties.

Most of the class and property definitions and integrity conditions stated in this document could be stated as RDF triples, using the RDF, RDFS and OWL vocabularies.

Full IRIs are cited in the text of this document in monospace font, enclosed by angle brackets, e.g., `<http://example.org/ns/example>`. Relative URIs are cited in the same way, and are relative to the base IRI `<http://example.org/ns/>`, e.g., `<example>` and `<http://example.org/ns/example>` are the same IRI. IRIs are also cited in the text of this document in an abbreviated form.

Abbreviated IRIs are cited in monospace font without angle brackets, and should be expanded using the table of abbreviations below.

| URI | Abbreviation |
|---|---|
| https://www.spire2030.eu/monsoon/ontologies/2017/09/csdm# | csdm |
| http://www.w3.org/1999/02/22-rdf-syntax-ns# | rdf |
| http://www.w3.org/2000/01/rdf-schema# | rdfs |
| http://www.w3.org/2004/02/skos/core# | skos |

**Table 3–List of URI abbreviations**

So, for example, `csdm:ProductionProcess` is an abbreviation of `<https://www.spire2030.eu/monsoon/ontologies/2017/09/csdm#ProductionProcess>`.

Most of the Cross-sectorial domain model concepts defines common documentation properties, which are formally defined in the following table.

| IRI | Domain/ Range/ Cardinality | Description |
|---|---|---|
| csdm:name | ObjectUnionOf(csdm:ProductionProcess, csdm:ProductionSegment, csdm:ProductionResource, csdm:DataElement, csdm:PhysicalDataElement, csdm:PhysicalDataSet, csdm:KPI, csdm:Model, csdm:PhysicalModel) xsd:string [1..1] | The name property specifies labels for the instances of the Cross-sectorial domain models, which are used to refer to them in natural language. Although identity of the instances is specified using the IRI representing RDF resource, readable names SHOULD BE unique at least within the scope of the one Production process. |
| csdm:description | ObjectUnionOf(csdm:ProductionProcess, csdm:ProductionSegment, csdm:ProductionResource, csdm:dependency, csdm:DataElement, csdm:PhysicalDataElement, csdm:PhysicalDataSet, csdm:KPI, csdm:Model, csdm:PhysicalModel) ObjectUnionOf(xsd:string,rdf:langString) [1..?] | The description property specifies unformal definition or description of the Cross-sectorial domain model instances in natural language. Description property can have assigned multiple RDF language literals with the localization of the description to multiple natural languages. |

**Table 4 – Common concept properties.**

## 5.1 Production process modelling

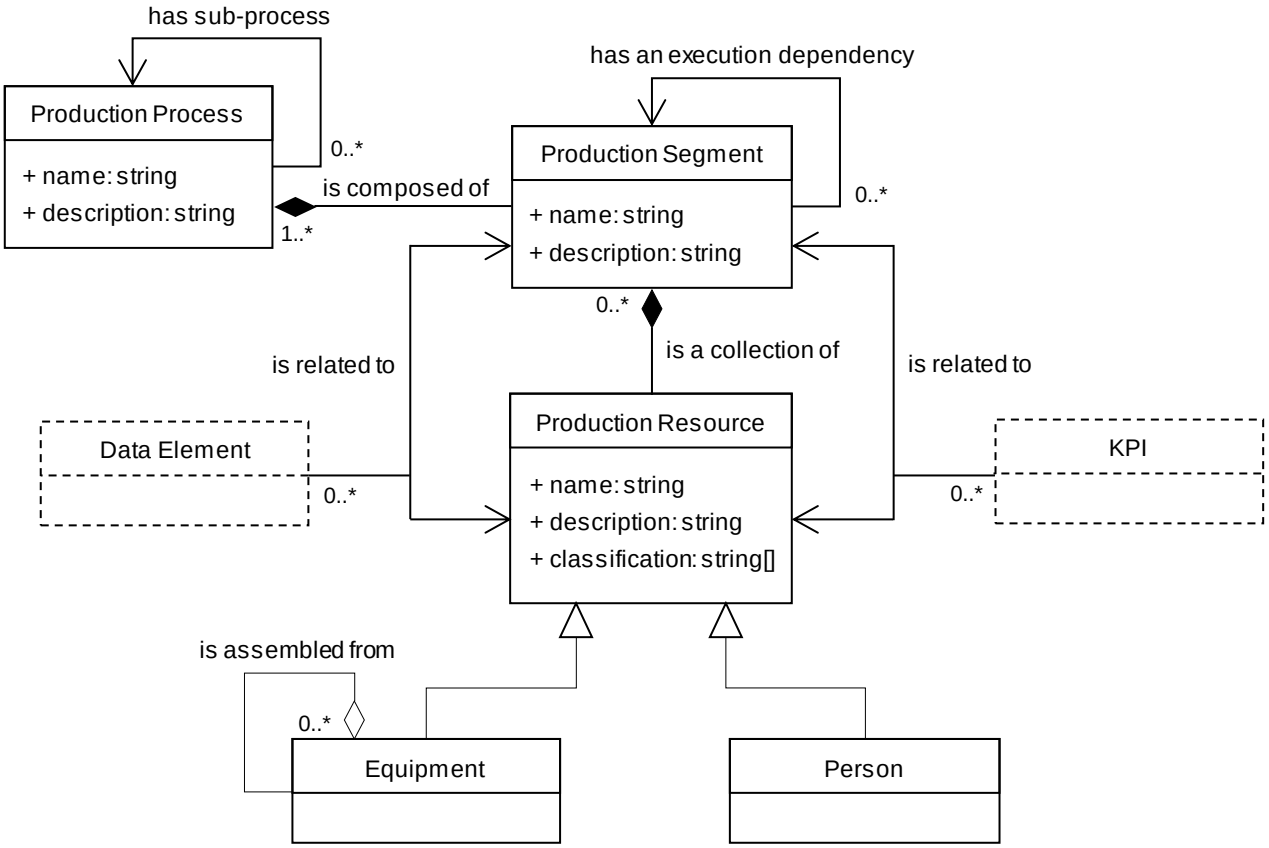


Figure 3 – Process modelling concepts.

### 5.1.1 Production Process and Production Segment

Production Process and Process Segments concepts represent decomposition of the overall production process into production steps. The concept of Process Segment represents logical grouping of the related Key-Performance Indicators, Data elements and Production resources (i.e. Equipment and human resources) required to carry out a production step. The model of Production Process is specified using the workflow graphical notation constructed from a limited set of graphical elements.

### 5.1.2 Production Resources

Production Resources are conceptually divided to Equipment resources and Person roles. Equipment concepts describe sites, areas, production units, production lines, work cells, process cells, or units linked to the specific Process Segment. Equipment may be hierarchically made up of several other sub-pieces of equipment (for example, a production line may be made up of work cells etc.), which is represented by "is assembled from" relation. Each may be defined as a separate equipment element with separate properties and capabilities. Additionally, a grouping of resources with similar characteristics and purposes can be described by Resource Classes. Any piece of equipment or any person role may be a member of zero or more classes. Initially, semantic model does not prescribe any taxonomy or schema for the Resource Classes and Production Resources can be annotated with arbitrary terms represented in the "classification" property.

| IRI | Superclass | Description |
|---|---|---|
| csdm:ProductionProcess | rdfs:Class | Production process is a container composed of Production segments (steps) ordered in the execution workflow.<br>Production process can be further decomposed to sub-processes. This can be represented by the compound production segment which delegates the execution to the underlying sub-process.<br>Concepts of the Production process type are modelled by graphical notation based on the simplified BPMN. |
| csdm:ProductionSegment | rdfs:Class | Process segment is a logical grouping of personnel and equipment resources required to carry out a production step. It defines what types of personnel and equipment are needed and it can describe specific resources (e.g. some specific type of equipment for particular segment).<br>Process segments are ordered in the execution workflow by execution dependency relation. Dependency relations are modelling using the graphical workflow notation based on the simplified BPMN. Internally, besides the RDF representation, the instances of the Production segment concept can be additionally represented as the objects for graphical notation specifying graphical attributes of the object such as shape, colour and position.<br>Each Process segment can be annotated with the relevant (logical) Data elements representing process segment inputs (e.g. quantities of consumed materials, control signals, etc.) and outputs (measurements from sensors, diagnostic signals). |
| csdm:ProductionResource | rdfs:Class | Production resource class is a common super-type for the concepts representing resources required to carry out a production step.<br>Grouping of resources with similar characteristics and purposes can be described by Resource Classes. Any piece of production resource may be a member of zero or more classes.<br>Each Production resource can be annotated with the relevant (logical) Data elements describing its parameters and capabilities. |
| csdm:Equipment | csdm:ProductionResource | Equipment concept cover sites, areas, production units, production lines, work cells, process cells, or units. Equipment may consist of other equipment, e.g. production line may be made up of work cells.<br>Each may be defined as a separate equipment element with separate properties and capabilities. |

| | | |
|---|---|---|
| | | Properties and capabilities are specified as the linked (logical) Data elements. |
| csdm:Person | csdm:ProductionResource | Person concept cover any human roles involved in the Production step and describes their capabilities. Capabilities are specified as the linked logical Data elements. |

Table 5–List of classes for Process modelling module.

| IRI | Domain/ Range/ Cardinality | Description |
|---|---|---|
| csdm:hasExecutionDependency | csdm:ProductionSegment csdm:ProuctionSegment [*..*] | Represents execution workflow order between the Production segments (steps). Besides the directed RDF property, this relation can have internal representation for graphical workflow notation specifying label and condition. |
| csdm:isAssembledFrom | csdm:Equipment csdm:Equipment [1..*] | Represents hierarchical decomposition of the Equipment to sub-pieces. Decomposition relations MUST BE constrained to acyclic hierarchy. |
| csdm:classification | csdm:ProductionResource ObjectUnionOf(xsd:string, skos:Concept) [*..*] | Specifies grouping of resources with similar characteristics and purposes. For the initial version, Cross-sectorial domain model do not prescribe any classification scheme for the resources, but defines Simple Knowledge Organization System (SKOS) as the preferred concept schema for specification of classification thesauri, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary. |

Table 6–List of properties and relations for Process modelling module.

### 5.1.3 Examples

The following example shows JSON-LD representation for one process segment (step) in the production process. As an example, we present the manufacturing process of green anode production, which consists of several sub-processes including milling, screening, proportioning and mixing of the grain, and vibrocompacting of the anode paste. Each of them (Vibrocompacting in our example) can be represented as a process segment including a description and specification of dependencies. Equipment can be specified to such process segment (in this case Vibrocompactor).

```
{
      "@id": "http://example.org/process1"
      "@type": "Process"
      "title": "Green Anodes Production"
      "description": "Process of manufacturing of green anodes consisting of
```

```
screening, classification and mixing of the grain into the paste and vibro-compaction of the paste."
}

{
      "@id": "http://example.org/Vibrocompacting1"
      "@type": "ProcessSegment"
      "title": "Vibrocompacting"
      "description": "Vibro-compaction is the technology employed for the formation of green carbon anodes. Produces well compacted anode block and tends to eliminate air bubbles and other factors that decrease the anode strength."
      "partOf": "http://example.org/process1"
      "hasExecutionDependency": ["Milling", "Screening"]
}

{
      "@id": "http://example.org/Vibrocompactor1"
      "@type": "Equipment"
      "title": "Vibrocompactor"
      "description": "Vibrocompactor is a machine consisting of a vibrating table, a mould where the anode paste is poured and a follower weight which helps to compact the paste"
      "classification": ["Compactor", "Hydraulic Machine"]
}
```


## 5.2 Data modelling

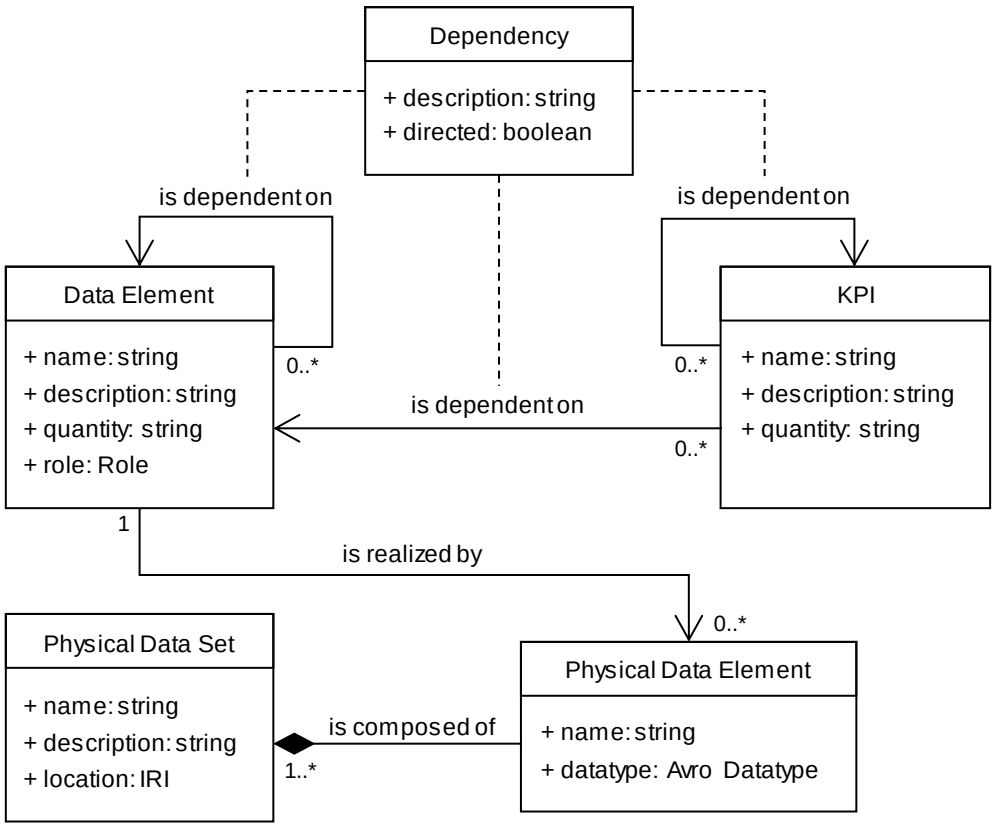


**Figure 4 – Data modelling concepts.**

### 5.2.1 Data Elements

Data elements are modelled in two levels of abstraction: logical and physical. Logical Data Element concept describes the role, type and quantity (e.g. units of measurements) of any data element related to the production step or equipment. Data element can have an input role (for measurements or input control signals) or output role (for diagnostic signals). Output data elements can be selected as a target attribute for the predictive function. One data element can have both roles depending on the goal of the data analysis (i.e. control signal can be an input for the predictive maintenance or output for the predictive control). The type of the data elements characterizes elements according to their values and denotes continuous, ordinal, nominal, spatial or series data elements.

Physical Data Elements link Logical Data Elements to the physical representation of the data in the structured records or files. Multiple Physical Data Elements (fields) can be grouped into one record or file and described by Physical Data Set. Each Physical Data Element has a specified data type. The data type can be primitive (e.g. byte, int, string) or complex (array, enumeration, map or union of types).

One simple yet effective specification for Data Elements is provided by the *Media Types for Sensor Measurement Lists* (SenML), which is a data model serializable with limited processing capabilities, able to fit large amount of data in a contextually representative format that is readable and writable by machines and humans.This specification defines media types for representing simple sensormeasurements and device parameters in the Sensor Measurement Lists(SenML) on the basis of *JavaScript Object Notation*(JSON), *Concise Binary Object Representation* (CBOR),*eXtensibleMarkup Language* (XML), and *Efficient XML Interchange* (EXI). Furthermore, Su et al. [10] have shown how to transform data expressed with SenMLinto the*Resource Description Framework* (RDF). Following the JSON representation, each Data Element is modelled according to the fields shown in the table below.

| Name | Label | Type |
|---|---|---|
| Base Name | bn | String |
| Base Time | bt | Number |
| Base Unit | bu | String |
| Base Value | bv | Number |
| Base Sum | bs | Number |
| Version | bver | Number |
| Name | n | String |
| Unit | u | String |
| Value | v | Number |
| String Value | vs | String |
| Boolean Value | vb | Boolean |
| Data Value | vd | String |
| Value Sum | s | Number |
| Time | t | Number |
| Update Time | ut | Number |

| | | |
|---|---|---|
| Link | l | String |

The SenML labels are used asthe JSON object member names within JSON objects representing theJSON SenML Records.The root JSON value consists of an array with one JSON object foreach SenML Record. All the fields in the above table may occur inthe records with member values of the type specified in the table.

An alternative to the SenML specification described above is the *OGC SensorThings API*standard specification [11]. This specification is an open standard for providing a unified way to interconnect *Internet of Things* (IoT) devices, data, and applications. The OGC SensorThings API comprises the sensing part, which provides a standard way to manage and retrieve observations and metadata from heterogeneous IoT sensor systems, and thetasking part, which is planned as a future work activity. The entities and their relationships of theOGC SensorThings API are shown in the following figure.

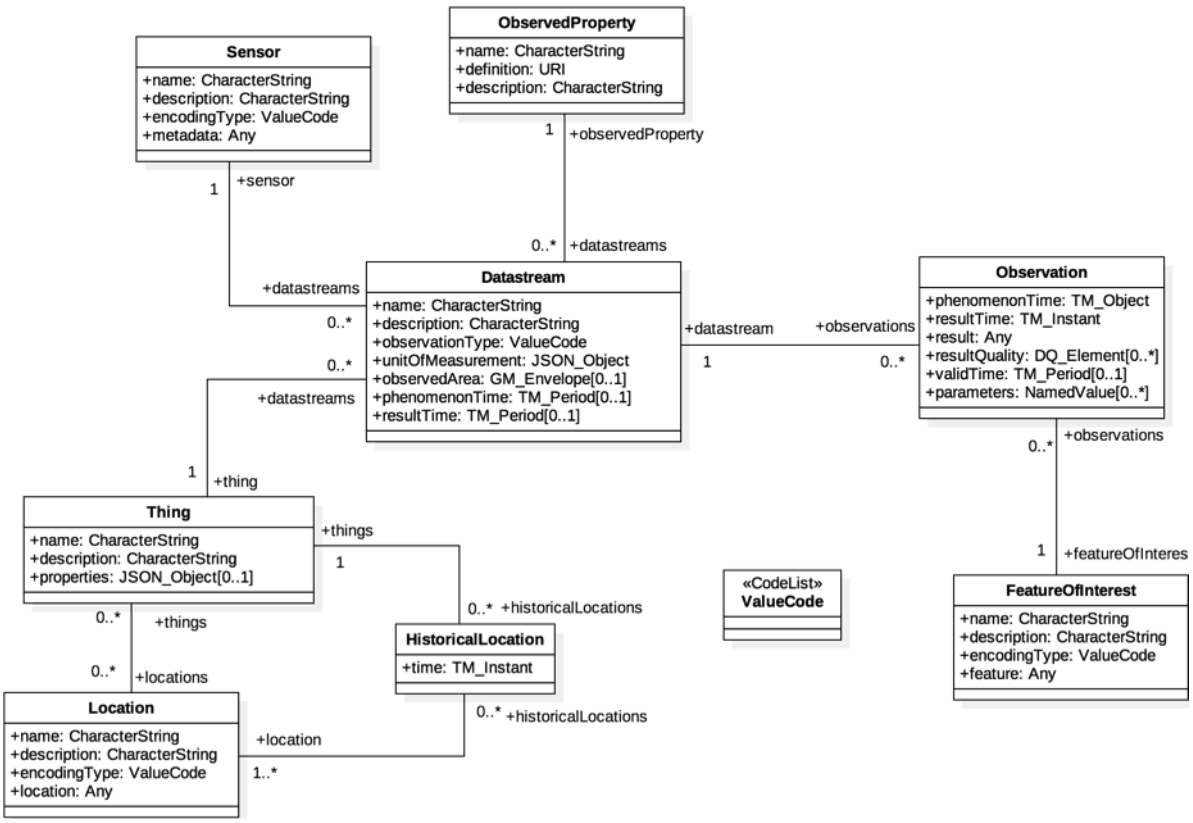


**Figure 5 – Illustration of the OGC SensorThings API**

As can be seen from the figure above, the OGC SensorThings API provides a high degree of flexibility in modelling Data Elements. Data Elements can be expressed as a Thing entity, which can either be physical, i.e. an object of the physical world, or virtual, i.e. some kind of information, or as a Sensor entity, which is an instrument that observes a property or phenomenon with the goal of producing an estimate of the value of the property. The Data Elements' quantities (e.g. units of measurements) are then captured by means of the Observation entity, which reflects the value of a property, together with the ObservedProperty entity, which might be used to specify the phenomenon of an Observation entity.

Both SenML specification and the OGC SensorThings API standard specification provide a JSON-based way of modelling Physical and Logical Data Elements and are thus to be further investigated within the Cross-sectorial Domain Model.

### 5.2.2 KPIs

The concepts of Key-Performance Indicators (KPIs) represent metrics designed to visualize, assess and manage the performance or impact of specific operations within the process industries. They are linked to the Process Segments or specified for overall Production Process. The main properties of the KPI metrics (e.g. type and quantity) are inherited from the Logical Data Elements (see the description in the previous subchapter). The KPIs can be grouped into KPI Categories according to the classification in ISO 2240.

Besides this classification, causal relations between KPIs can be expressed by the KPI Dependency relations, which transform one KPI to another (e.g. material or energy savings can be converted into KPIs for the environmental impact such as level of emissions). The input of the KPI Dependency can be also Evaluation Results, which estimate performance of the predictive functions (see the description of the predictive functions in the subsequent subchapter). Using the composition of KPI Dependency relations, it is possible to infer impact of the deployment of a predictive function on the performance of the production process, where the performance can be evaluated from various perspectives such as material/energy consumption, product quality, environmental impact, etc.

| IRI | Superclass | Description |
|---|---|---|
| csdm:DataElement | rdfs:Class | Logical Data element describes any datum which can be measured, observed or set as the control parameter during the production process. It defines data at the logical level without any details dependent ondata format or how data are acquired, transformed or represented in the information systems.<br>Data element can have an input role (for measurements or input control signals) or output role (for diagnostic signals). Output data elements can be selected as a target attribute for the predictive function. One data element can have both roles depending on the goal of the data analysis (i.e. control signal can be an input for the predictive maintenance or output for the predictive control). |
| csdm:PhysicalDataElement | rdfs:Class | Physical Data element concept specifies details how the logical Data element is represented in the information systems. One Data Element can be mapped to multiple Physical Data Elements depending on the data pre-processing methods applied on the data during the process of data acquisition and integration. Physical data elements can be grouped to the Physical Data sets.<br>Physical data element specifies format of data which is defined using the Avro datatype scheme. |
| csdm:PhysicalDataset | rdfs:Class | Physical Data set is the collection of Physical Data elements. It corresponds to the one relational table, collections of objects in NoSQL database or flat |

| | | |
|---|---|---|
| | | structured file with the tabular or transaction data. The physical location of the data set is represented as the IRI. For Physical Data sets, it is possible to infer datatype schema (represented as the composed Avro datatype), which is the composition of datatypes of the grouped Physical Data elements. |
| csdm:Dependency | rdfs:Class | This concept represents properties of dependency retaliation between two logical Data Elements or KPIs or between the KPI and Data Element. The Dependency relation can be directed or undirected, where the direction of the relation represents causation anticipated between the elements.<br>In the initial Cross-sectorial domain model specification, the functional relation between the elements will be described informally in natural language using the `csdm:description`property. |
| csdm:KPI | rdfs:Class | The concepts of Key-Performance Indicators (KPIs) represent metrics designed to visualize, assess and manage the performance or impact of specific operations within the process industries. They are linked to the Process Segments or specified for overall Production Process.<br>Technically, KPI has to be measurable so the concept semantically share some properties with the Data elements such as `csdm:quantity`.<br>It is possible to specify dependencies between the KPIs and Data elements, i.e. one KPI can be influenced or computed from another KPI (or multiple KPIs) or from measurable Data elements. |

**Table 7–List of classes for Data modelling module.**

| **IRI** | **Domain/ Range/ Cardinality** | **Description** |
|---|---|---|
| csdm:from | csdm:Dependency<br>ObjectUnionOf(csdm:DataElement, csdm:KPI)<br>[1..*] | Represents source Data element of the Dependency relation. In the case of directed dependency, `csdm:from`denotes independent element. |
| csdm:to | csdm:Dependency<br>ObjectUnionOf(csdm:DataElement, csdm:KPI)<br>[1..*] | Represents target Data element of the Dependency relation. In the case of directed dependency, `csdm:to`denotes dependent element. |
| csdm:directed | csdm:Dependency<br>xsd:boolean<br>[1..1] | Indicates if the dependency relation is directed or undirected. Directed dependencies represents causation anticipated between two data elements or KPIs. |

| | | |
|---|---|---|
| csdm:isRealizedBy | csdm:DataElement<br>csdm:PhysicalDataElement<br>[1..*] | This relation links logical Data element to Physical Data element. One logical Data element can be realized by multiple Physical Data elements, since the same quantity can be processed and stored using the different data formats in different datasets. |
| csdm:isComposedOf | csdm:PhysicalDataSet<br>csdm:PhysicalDataElement<br>[1..*] | Groups together multiple Physical Data elements into data collection represented by Physical Data set concept. |

**Table 8–List of properties and relations for Data modelling module.**

### 5.2.1 Examples

In this example, we consider the Data Element containing the power consumption data of Buss Mixer involved in process of green anodes production. The example illustrates the element description and its binding to particular physical data represented by Physical Data Element. Data element specifies the quantity –measurement units. Then, a dependency between the measurements and KPI representing the energy consumption of the device is defined.

```
{
      "@id": "http://example.org/BUSS-mixer-power"
      "@type": "DataElement"
      "title": "Mean mixer power"
      "description": "Data element containing the Mean mixer power values."
      "quantity": "kW"
      "role": "input"
      "isRealizedBy": "http://example.org/D110-J160_PUISSANCE_MOY_MALAXEUR"
}

{
      "@id": "http://example.org/EnegryConsumption1"
      "@type": "KPI"
      "title": "Energy consumption of the device."
      "description": "Overall electrical energy consumption of the production
process."
      "quantity": "kW/h"
}

{
      "@type": "Dependency"
      "description": "Higher mean mixer power leads means higher power
consumption."
      "from": "http://example.org/BUSS-mixer-power"
      "to": "http://example.org/EnergyConsumption1"
      "directed":true
}

{
      "@id": "http://example.org/D110-J160_PUISSANCE_MOY_MALAXEUR"
      "@type": "PhysicalDataElement"
      "title": "D110-J160_PUISSANCE_MOY_MALAXEUR"
      "description": "Values of the mean mixer power, time series, acquisition
frequency 5s."
      "datatype": "float"
      "partOf": "http://example.org/dataset1"
```

}

## 5.3 Predictive functions modelling

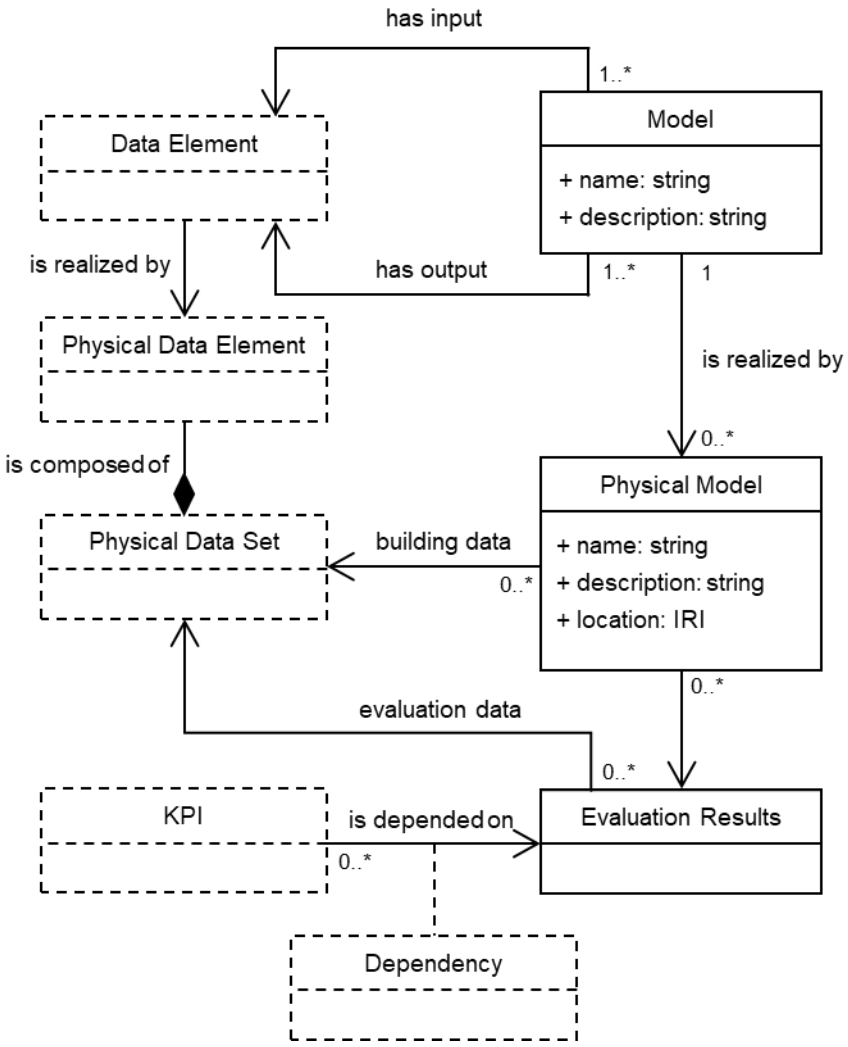


Figure 5– Predictive functions modelling concepts.

### 5.3.1 Predictive functions

Similarly to Data Elements, predictive functions are modelled at the physical and logical level. The concept of predictive function Model specifies which data elements are the inputs to the predictive function (i.e. predictors or independent variables) and which data elements are the outputs (i.e. dependent or predicted variables). Additionally, Physical Model specify details about the building/training data and algorithm used for building a specific predictive function and Evaluation Results describing the performance of the predictive function evaluated on the validation set or real-time operational data.

| IRI | Superclass | Description |
|---|---|---|
| csdm:Model | rdfs:Class | Model concept represents logical definition of the predictive function. It specifies, which logical Data elements are expected as the input attributes (independent variables) |

| | | |
|---|---|---|
| | | of the predictive function and what is the output predicted value (target attributes or dependent variables).The role of the logical Data model is to specify goal of the data analysis without the implementation details how the physical data have to be integrated and pre-processed in the particular information system environment and for the particular data analysis method. |
| csdm:PhysicalModel | rdfs:Class | Physical model represents predicted function build on the particular training Physical Data set by selected data analysis method or algorithm. One logical Data model can be linked to/realised by multiple Physical Data models, which can differ in the building (training) Data set, algorithm or algorithm parameters set by the data scientist. Each Physical Model can have also different quality evaluated on the evaluation (testing) Data set. |
| csdm:EvaluationResults | rdfs:Class | This concept represents results of the quality evaluation of the particular physical predictive function. Quality of the predictive functions are usually evaluated by applying the predictive function and computing of quality metrics on the validation (testing) Data set, which is independent to the build (training) data. The concept can be represented as the container for estimated metrics (such as prediction accuracy, which measures the ratio of correct predictions to the total number of cases evaluated). |

**Table 9–List of classes for Predictive functions modelling module.**

| IRI | Domain/ Range/ Cardinality | Description |
|---|---|---|
| csdm:hasInput | csdm:Model csdm:DataElement [*..*] | Specifies input of the logical predictive function (i.e. independent variable). |
| csdm:hasOutput | csdm:Model csdm:DataElement [*..*] | Specifies output of the logical predictive function (i.e. dependent variable). |
| csdm:isRealizedBy | csdm:Model csdm:PhysicalModel [1..*] | This relation links logical predictive function to physical model build on the training data. One logical predictive function can be realized by multiple Physical models, since the same predictive function can be built on the different set of training data or by applying different algorithms or algorithm settings. |
| csdm:buildingData | csdm:PhysicalModel csdm:PhysicalDataSet [*..+] | This relation links Physical model with the training Data set used to build the model. |
| csdm:evaluationData | csdm:EvaluationResults csdm:PhysicalDataSet | This relation links Evaluation results with the testing Data set used for the model evaluation. |

| | | |
|---|---|---|
| | [*..+] | |

Table 10–List of properties and relations for Predictive functions modelling module.

### 5.3.2 Examples

This example presents the representation of how the concrete predictive function. In this case, model for prediction of anode batch quality is realized by physical Random-forest classification model. That model is built using the specified data which refer to the anode sensor physical data set located in the HDFS distributed storage.

```
{
      "@id": "http://example.org/predictive-function1"
      "@type": "Model"
      "title": "Prediction of anode batch quality"
      "description": "Model for predicting of quality of batches of anodes
produced during 8h shifts. Target attributes has 2 values representing low and
good quality."
      "isRealizedBy": ["http://example.org/model-random-forest1"]
}

{
      "@id": "http://example.org/model-random-forest1"
      "@type": "PhysicalModel"
      "description": "Random forest ensemble classification model."
      "buildingData": "http://example.org/dataset1"
}

{
      "@id": "http://example.org/dataset1"
      "@type": "PhysicalDataSet"
      "title": "Anode sensor data"
      "description": "Sensor data containing from anode density info,
dimensions, coke quality, production time, overall 46 attributes."
      "location": "hdfs://data/examples/dataset1.csv"
}
```

## 6 Evaluation of the semantic models

For the evaluation of the semantic models during the design, we have adopted methodology proposed by Grüninger and Fox [8]. The methodology proposes to first define requirements in the form of questions that semantic model must be able to answernamed ontology competency questions. In the next step, the terminology of the semantic model and its classes and relations should be defined. The competency questions in this phase are informal, since they are still not expressed in the formalized language of the ontology. Once the informal competency questions have been posed for the new ontology, then the terminology of the ontology (its classes and relations) are specified using some first order logical language. This language must provide the necessary terminology to formally restate the informal competency questions. Every newly developed ontology must be accompanied by a set of formal competency questions. This represents a way how to evaluate the ontology and claim that it is adequate.Finally, ontologies can be distinguished by the competency questions which they are capableof answering.

For the case of the Cross-sectorial Domain Model ontology, presented in this deliverable, we established for all three modules a list ofinformal competency questions in the design phase.Having builtall three ontology modules and expressed them in a formal ontology language based ondescription logics, we can formulate the formal competency questions for eachof the modules and show that the ontology module is capable of

solving and answeringthe questions it is designed to answer.For that purpose, we formulate the questions usingthe SPARQL query language. All query examples assume common prefixes for URL specified in Table 11.

| ID | Informal query | SPARQL query |
|---|---|---|
| 1 | Which data elements are related to the given production segment? | SELECT ?element WHERE {<br>?elementrdf:typeDataElement.<br>?elementcsdm:isRelatedTo<segment ID>.} |
| 2 | Which key-performance indicators are relevant to the given production segment or for the overall production process? | SELECT ?kpi WHERE {<br>?kpirdf:type KPI.<br>?kpicsdm:relatedTo ?id.<br>?idrdf:type ?type.} FILTER(<br>?type IN (ProductionSegment, ProdutionProcess)) |
| 3 | Which predictive function can be applied to optimize the given process segment? | SELECT ?function WHERE {<br>?functioncsdm:output ?element.<br>?elementcsdm:relatedTo<segment ID>.} |
| 4 | Which predictive function can be applied to optimize the given key-performance indicator? | SELECT ?function WHERE {<br>?functioncsdm:output ?element.<br>?depcsdm:source ?element.<br>?depcsdm:target<KPI ID>.} |
| 5 | Which data elements are influencing the given key-performance indicator? | SELECT ?element WHERE {<br>?depcsdm:source ?element.<br>?depcsdm:target<KPI ID>.} |
| 6 | Which data elements are influencing the given data element? | SELECT ?element WHERE {<br>?depcsdm:source<element ID>.<br>?depcsdm:target ?element.} |
| 7 | What is the impact of the predictive function quality to the given key-performance indicator? | SELECT ?dep WHERE {<br>?depcsdm:source ?stats.<br>?depcsdm:target<KPI ID>.<br>?statscsdm:relatedTo<function ID>.}<br><br>The current specification just describe dependency as the test. Next update will specify machine-readable functional description. |
| 8 | Which data sets can be used to build the given predictive function? | SELECT ?dataset WHERE {<br>?datasetcsdm:isComposedOf?physelm.<br>?elementcsdm:isRealizedBy ?physelm.<br>{<function ID>csdm:input ?element} UNION<br>{<function ID>csdm:output ?element}} |
| 9 | How to validate that all data elements required to apply predictive function are available? | It can be implemented as the series of constraint queries in the form:<br><br>SELECT ?physelm WHERE {<br>?elmcsdm:isRealizedBy ?physelm<br><function ID>csdm:input ?element.}<br><br>It has to be a binding to physical element (i.e. not |

| | | |
|---|---|---|
| | | empty result) for all logical inputs of the predictive functions. |

**Table 11–List of competency questions for design and evaluation of the Cross-sectorial Domain Model.**

# 7 Conclusions

This deliverable describes the initial specification of the Cross-sectorial domain model. The main part of the document describes normative formal specification and machine-readable format for storing and exchanging of Cross-sectorial domain models. The machine-readable format will be complemented by the specification of the human-readable form and graphical notation in the deliverable D4.1 Initial Semantic framework as the part of the design of Semantic framework modelling tools. This initial specification includes all main concepts required for modelling of production processes, data elements and predictive functions.In the deliverable D2.8 Final Cross-sectorial domain model, definition of the concepts will be extended with the new properties and relations, where we will incorporate our experiences from the modelling of pilot aluminium and plastic domains and extend scope of the models to modelling of the functional dependencies between data elements and KPIs.

## Acronyms

| Acronym | Explanation |
|---|---|
| B2MML | Business To Manufacturing Markup Language |
| CRISP-DM | Cross-industry Standard Process for Data Mining |
| DMG | Data Mining Group |
| ERP | Enterprise Resource Planning |
| JSON | JavaScript Object Notation |
| JSON-LD | JavaScript Object Notation for Linked Data |
| KPI | Key Performance Indicator |
| MESA | Manufacturing Enterprise Solutions Association |
| OWL | Web Ontolodgy Language |
| PFA | Portable Format for Analytics |
| PMML | Predictive Model Markup Language |
| RDF | Resource Description Framework |
| RDF-S | Resource Description Framework Schema |
| SEMMA | Sample, Explore, Modify, Model, and Assess |
| SPARQL | SPARQL Protocol and RDF Query Language |
| UML | Unified Modeling Language |
| URI | Unified Resource Idenifier |
| W3C | World Wide Web Consortium |
| XML | Extensible Markup Language |

## List of figures



## List of tables

## References

[1] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "Crisp-Dm 1.0," *CRISP-DM Consortium*, p. 76, 2000.

[2] U. Shafique and H. Qaiser, "A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA )," *International Journal of Innovation and Scientific Research*, vol. 12, no. 1, pp. 217–222, 2014.

[3] F. Manola, E. Miller, and B. McBride, "RDF primer," *W3C recommendation*, vol. 10, no. February 2004, pp. 1–107, 2004.

[4] D. Brickley and R. V. Guha, "RDF Schema 1.1 - W3C Recommendation," *World Wide Web Consortioum*, 2008. [Online]. Available: http://www.w3.org/TR/rdf-schema/.

[5] M. Sporny, G. Kellogg, and M. Lanthaler, "Json-Ld 1.0," *A JSON based Serialization for Linked Data*, no. January, pp. 1–33, 2013.

[6] L. S. Gould, "B2MML Explained.," *Automotive Design & Production*, vol. 119, no. 2, p. 54, 2007.

[7] R. Pechter and Rick, "What's PMML and what's new in PMML 4.0?," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, p. 19, Nov. 2009.

[8] J. Pivarski, C. Bennett, and R. L. Grossman, "Deploying Analytics with the Portable Format for Analytics (PFA)," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016, pp. 579–588.

[9] M. Grüninger, M. S. Fox, and M. Gruninger, "Methodology for the Design and Evaluation of Ontologies," *International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 1–10, 1995.

[10] Xiang Su, Hao Zhang, JukkaRiekki, Ari Keränen, Jukka K. Nurminen, Libin Du: "*Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF*". ANT/SEIT 2014: 215-222

[11] Liang, Steve H.L., Chih-Yuan Huang, and Tania Khalafbeigi. "*OGC SensorThings API Part I:Sensing*" OGC® Implementation Standard (2016)