PROPHESY

Platform for rapid deployment of self-configuring and optimized predictive maintenance services

DELIVERABLE

D3.3 – Digital Modelling and Interoperability v1

Horizon 2020
European Union funding
for Research & Innovation

The PROPHESY initiative has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 766994.

Project Acronym:	PROPHESY
Grant Agreement number:	766994 (H2020-IND-CE-2016-17/H2020-FOF-2017)
Project Full Title:	Platform for rapid deployment of self-configuring and optimized predictive maintenance services
Project Coordinator:	INTRASOFT International SA

This project is co-funded by the European Union

DELIVERABLE

D3.3 – Digital Modelling and Interoperability v1

Dissemination level	(PU) – Public
Type of Document	(R) – Report
Contractual date of delivery	M12, 30/09/2018
Deliverable Leader	AIT
Status - version, date	Final – v1.0, 04/10/2018
WP / Task responsible	WP3 (MONDRAGON)/Task 3.2 (AIT)
Keywords:	Data Models, MIMOSA, Interoperability Registry, Digital Twins, Automation, Predictive Maintenance

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 766994.

Executive Summary

Predictive maintenance systems are usually based on the development of predictive analytics applications that are developed and executed over very large volumes of maintenance related data, which stem from a wide array of different data sources. As such they have to deal with the collection, consolidation and semantic unification of many different data sets of diverse that types that are produced by many different sources. The PROPHESY-CPS platform aims at facilitating the data collection and consolidation process using a set of common standards-based digital models in order to:

- Allow interoperability across the different data collection systems and across the diverse datasets that their produce. This includes semantic interoperability of the different datasets in order to allow their consolidated processing, interpenetration and use.
- Exchange of maintenance related datasets across different instances of PROPHESY deployments such as deployments residing within different factories of a company or even within different supply chain partners.
- Ensuring that the various PROPHESY applications (e.g., analytics applications in WP4 and visualization applications in WP5) need to employ a single set of data preparation and data transformation processes regardless of the diversity of sensors and other data collection systems.

This deliverable describes the PROPHESY approach to digital modelling for predictive maintenance and presents the specifications of the PROPHESY data models (PROPHESY-DM). It is a report on the specification of the digital models, yet it also provides insights on the prototype implementation of the models as part of the PROPHESY-CPS platform. The latter prototype implementation is based on the reuse of models and databases developed in other projects of the PROPHESY partners. The approach followed is based on an interoperability registry infrastructure, which provides the means for linking object-level information from different databases and repositories that include data and metadata about a specific object. In this way, the PROPHESY-DM infrastructure is flexibly extensible with additional data from other repositories, while minimizing the transformations and middleware developments needed to integrate new repositories and their data sources. In particular, integration of a new repository is based on a simple linking of the objects it contains to the respective objects of the interoperability registry and does not require complex middleware-based transformation.

The PROPHESY-DM infrastructure comprises the following main components:

- A database comprising observations from streaming data sources that can be used for predictive maintenance. The design and implementation of this databases has been based on the digital modelling approach of the H2020 FAR-EDGE project, which provides the means for representing and configuring streaming data sources in industrial environments.
- A database comprising maintenance related metadata. This database has been implemented over the database of the MANTIS project, which leverages MIMOSA standards. It is essentially a lightweight MIMOSA database, which contains maintenance specific metadata to support the PROPHESY applications.

- An Interoperability Registry (IR), which provides the means of linking different repositories at the object level. It is implemented based on the Open O&M Common Interoperability Registry (CIR).

As evident from the above-listed descriptions, all three components are implemented over existing components and technologies of the project partners. In addition to these databases, a management console will be implemented in order to ease management and configuration of the PROPHESY-DM infrastructure and of all repositories linked to it.

The specification and implementation of the PROPHESY-DM infrastructure has been driven by the data integration and interoperability requirements of the project. Moreover, the background technologies of the partners and their capabilities have been considered. Therefore, the deliverable illustrates the requirements that influenced the PROPHESY-DM infrastructure design, along with standards and technologies that were considered as part of the implementation.

Note that the data modelling infrastructure of this deliverable is the first version of the PROPHESY data modelling outcomes. This version will be used in the scope of the PROPHESY-CPS platform and will be integrated in various PROPHESY applications, notably visualization and analytics applications. Based on this integration we aim at validating our claims for the functionality and extensibility of the infrastructure. At the same time, we expect to receive validation and evaluation feedback, which will be taken into account towards producing the final version of the PROPHESY-DM infrastructure. The latter will be described in deliverable D3.4, which can be seen as the next version of the present deliverable.

Deliverable Leader:	Athens Information Technology (AIT)
Contributors:	AIT
Reviewers:	SENSAP, INTRA
Approved by:	INTRA

Document History			
Version	**Date**	**Contributor(s)**	**Description**
0.1	28/03/2018	AIT	Initial Structure for Discussion in WP3 Telco
0.11	24/07/2018	AIT	Revisions to the Structure during San Sebastian Meeting
0.12	02/08/2018	AIT	Section 1
0.15	21/08/2018	AIT	Initial inputs to Section 2
0.20	31/08/2018	AIT	Authoring of Section 3 – Inputs on Standards and Relevant Projects (PERFORM, FAR-EDGE)
0.25	04/09/2018	AIT	Initial Inputs in Section 4 about the PROPHESY approach to Data Modelling
0.30	13/09/2018	AIT	Updates in Section 4, completion of Section 3
0.40	25/09/2018	AIT	Updates in the schemas in Section 4
0.45	28/09/2018	AIT	Alignment of the Digital Modelling work to the PROPHESY-CPS architecture developments
0.50	02/10/2018	AIT	Various edits and preparation of version for Quality Review
1.0	04/10/2018	AIT	Final Version

Table of Contents

Table of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AM	Analytics Manifest
APD	Analytics Processor Definition
API	Application Programming Interface
APM	Analytics Processor Manifest
B2MML	Business to Manufacturing Markup Language
CAEX	Computer Aided Engineering Exchange
CBM	Condition-Based Maintenance
CCOM	Common Conceptual Object Model
CIR	Common Interoperability Registry
CMM	Computerized Maintenance Management System
CRIS	Common Relational Information Schema
CPS	Cyber Physical System
DDA	Distributed Data Analytics
DI	Data Interface
DK	Data Kind
DM	Digital Model
DSD	Data Source Definition
DSM	Data Source Manifest
ERP	Enterprise Resource Planning
FMECA	Failure Mode, Effects, and Criticality Analysis
IR	Interoperability Registry
KPI	Key Performance Indicator
MES	Manufacturing Execution System
ML	Machine Learning
OSA-CBM	Open System Architecture for Condition-Based Maintenance
OSA-EAI	Open System Architecture for Enterprise Application Integration
PDM	Predictive Maintenance
PERFoRMML	PERFORM Project Markup Language
PO	Processor Orchestrator
QARMA	Quantitative Association Rule Mining Algorithms
RDBMS	Relational Database Management Systems
RUL	Remaining Useful Life
REST	Representational State Transfer
SOA	Service Oriented Architecture
XML	Extensible Markup Language

1 Introduction

1.1 Data Modelling and Interoperability in PROPHESY

PROPHESY is developing and validating a novel platform (i.e. PROPHESY-PDM) for predictive maintenance solutions, which is destined to lower the effort and cost needed to develop and deploy predictive maintenance solutions in industrial plants. A core element of the PROPHESY platform is its Cyber-Physical platform part (i.e. the PROPHESY-CPS platform), which deals with the collection of maintenance-related data from various data sources, including sensors, cyber-physical systems, industrial data collection platforms and business information systems. The collection of such data is a key prerequisite for the implementation of condition-based maintenance systems (including predictive maintenance), given that the extraction of predictive maintenance insights and related recommendations hinges on the analysis of maintenance related datasets. Therefore, beyond the collection of maintenance-related datasets from various sources, the PROPHESY-CPS platform should provide the means for consolidating these datasets and unifying their semantics, as means of facilitating their integrated processing. Overall, PROPHESY-CPS must facilitate the dynamic integration of diverse sources of maintenance data in the PROPHESY-PDM solutions, including the harmonization of their semantics with the semantics of other sources.

One of the main vehicles for the semantic unification of maintenance data from different data sources is the transformation of the formats and semantics of the various data sources to a proper unified data model. Such a data model could be based on standard-based data models for condition monitoring and maintenance application in industrial environment. Adherence to an agreed standards-based model can serve as basis for interoperability across different data sources, as well as for unified processing of maintenance data regardless of the data sources where they belong. Moreover, the use of standards-based models can also facilitate data sharing and interoperability across different maintenance systems, such as two different PROPHESY-PDM deployment instances. In this direction, the specification of proper data models for representing maintenance datasets, should be accompanied by the development of appropriate middleware libraries for accessing data formatted according to the common data models, including APIs (Application Programming Interfaces) and tools that facilitate CRUD (CReate Update Delete) of data elements in instances of these data models.

The present deliverable is devoted in the illustration of the data models that will be used in PROPHESY, but also of the infrastructure that will enable PROPHESY applications to access data formatted according to these data models. The latter infrastructure is conveniently called PROPHESY-DM (Data Models) infrastructure in later sections of the deliverable. In principle, PROPHESY specifies and leverages platform and vendor agnostic digital models for representing maintenance datasets as part of the PROPHESY-CPS platform. The models are presented in the form of proper data schemas (e.g., XML Schemas), which will capture the maintenance data sources and operations. As part of the deliverable we also present the background and rationale behind the selection and specification of the PROPHESY data models. The latter selection and specification has been driven by a range of different factors

including the intention to reuse results from background projects of the partners, the need to comply with standards, as well as the need to cover both maintenance and automation use cases. Hence, the deliverable presents also the scope and purpose of digital modelling in PROPHESY, which is not limited to facilitating interoperability, but rather supports a range of configuration and digital simulation (i.e. "digital twins") tasks as well.

Apart from reporting on the structure and the schemas of the various PROPHESY data/digital models, the deliverable presents the technologies and techniques that are used to implement data access and CRUD operations over databases structured based on the PROPHESY data models. According to the project's DoA, this deliverable is destined to be a report. Nevertheless, the specifications given in the report are destined to be implemented as part of the PROPHESY-CPS implementation. Therefore, we have opted to provide the implementation details of the PROPHESY Data Models infrastructure.

Note that the present deliverable represents the first version of the project's digital modelling and interoperability approach. The second and final version of PROPHESY's digital modelling and interoperability deliverable (i.e. deliverable D3.4) will be an enhanced version of the present document, including the final specification of the PROPHESY digital models, along with more functionalities and fine tuning based on stakeholders' feedback about the use of the results and prototypes of the first version.

1.2 Methodology

The methodology for specifying the PROPHESY Digital Models involved the following activities, which are illustrated in Figure 1 and are described in detail in the following paragraphs of this deliverable:

- **Identification of Digital Models Purpose and Use Cases**: Digital modelling in digital manufacturing and industrial maintenance is a quite broad topic, as digital models can be used for a wide range of different tasks. Therefore, prior to specifying the PROPHESY digital models we have identified the scope of the digital modelling task in PROPHESY and how digital models can be used to serve the PROPHESY objectives. Along with these use cases, we have also identified the main functional and non-functional requirements that should be fulfilled by the PROPHESY Digital Models.
- **Review and Analysis of Existing Digital Models**: We have also reviewed and analysed existing digital models (including standards-based digital models) in support of our digital modelling task. The main objective of this analysis was to identify readily available models and modelling concepts that could be reused in order to meet the PROPHESY requirements. In our case reuse led to a faster bootstrapping of the digital models specification and implementation. Note that our digital models review and analysis effort has been kept minimal, as we have exploited the results and outcomes of similar reviews that have been recently conducted in project where some of the partners participate, including the H2020 FAR-EDGE and H2020 PERFORM projects.

- **Selection and Specification of PROPHESY Digital Models**: Taking into account existing digital models and the PROPHESY digital modelling requirements, an initial specification of the PROPHESY Digital Models for interoperability has been provided. To this end, a confrontation of the properties of existing models against PROPHESY requirements and use cases has been provided.

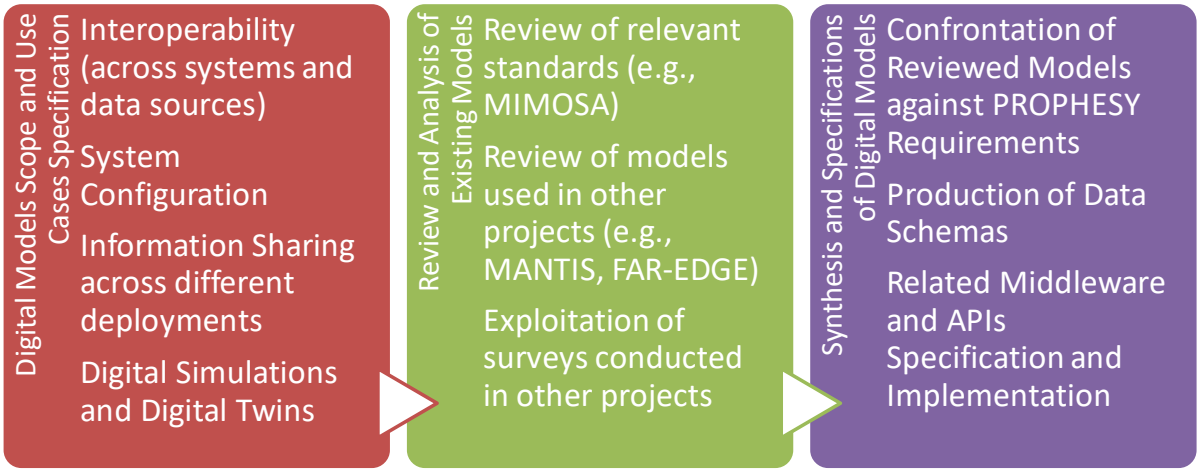


Figure 1: PROPHESY Digital Modelling Methodology and Relevant Activities

1.3 Relationship to other deliverables

Several PROPHESY subsystems and modules deal with maintenance data, being either data producers or data consumers in the scope of the PROPHESY-PDM platform. Hence, the present deliverable is closely affiliated to a range of other deliverables of the project, notably the deliverables that specify and/or implement consumers or producers of maintenance data. The most relevant deliverables to the present one are:

- **D3.1 PROPHESY-CPS Detailed Architecture (V1)**, which specifies the architecture of the PROPHESY-CPS platform, which shall include the specification of the means for ensuring interoperability based on digital modelling. Overall, the PROPHESY digital models specification and middleware mechanisms should be aligned to the architecture of the PROPHESY-CPS platform, given that digital modelling middleware is essentially part of PROPHESY-CPS.
- **D3.5 PROPHESY Sensor Data Collection (V1)**, which should interface with digital models middleware mechanisms in order to ensure that sensor data collection provides the means for data unification and interoperability regardless of the type and format of the sensor data source.
- **D3.7 PROPHESY-CPS Middleware Infrastructure (V1),** which should be streamlined with the PROPHESY digital modelling middleware and CRUD mechanisms.
- **D4.1 Automatic Data Collection (V1)**, which shall specify and implement the way data will be consumed by the machine learning and data analytics modules of the PROPHESY platform (PROPHESY-ML). This deliverable is expected to deal with the data consumption side, while D3.5 is more inclined towards the production side.

1.4 Document Structure

The current document is structured as follow:

- Section 2 following this introduction presents the scope and importance of digital models for PROPHESY. It illustrates the main uses of digital models within and outside PROPHESY, along the with the main requirements that drive the PROPHESY digital models specifications.
- Section 3 reviews existing digital models for industrial automation and maintenance. It also audits them against PROPHESY requirements and use cases.
- Section 4 presents the specification of the PROPHESY digital models with reference to a collection of schemas that cover both streaming data modelling and maintenance metadata. The latter schemas are provided as an Appendix in this document. Moreover, Section 4 outlines the implementation details of the PROPHESY-DM infrastructure, including the technologies and tools to be used.
- Section 5 is the final and concluding section of the deliverable. It draws main conclusions and provides an outlook for the next and final version of the deliverable.

2 Driving Requirements and Scope of Data Modelling

In this section we underline the need for digital modelling in PROPHESY, based on the presentation of possible use cases where digital models add value. Moreover, we outline some requirements regarding the specification and implementation of digital models, which will be taken into account in the selection and implementation of the PROPHESY models in Section 4.

2.1 Motivating Use Cases for Digital Modelling and Interoperability

2.1.1 Maintenance Information Storage and Representation

A PROPHESY system for predictive maintenance relies on the collection and analysis of digital data about the condition of machines and equipment. Following the collection of such information, it is mandatory to represented in a structured digital format (i.e. through an appropriate digital model) in order to facilitate its storage in a database and its processing by analytics algorithms, such as algorithms that are part of the PROPHESY-ML toolbox. Overall, a digital model specifies the data schema for:

- Representing maintenance related data in the digital world.
- Persisting the data in a data-store.
- Processing the data based on some machine learning or data mining algorithm.

In this context, the use of a common standards-based digital model for maintenance data in PROPHESY can provide the following advantages:

- It facilitates data scientists in their pre-processing and analytics task, as they have to deal with a single rather than multiple data formats and associated sets of semantics.
- It facilitates the storage and management of maintenance datasets through hiding the complexity and heterogeneity of the various data sources.

The concept is overall depicted in Figure 2, which illustrates why data pre-processing for analytics applications (e.g., applications of the QARMA algorithms that are used in PROPHESY WP4 or other digital twins applications) is made simple. The figure illustrates also that a middleware infrastructure that transforms data from their source format to the PROPHESY digital models format can be used to ensure adherence of different data sources to a standards-based format.

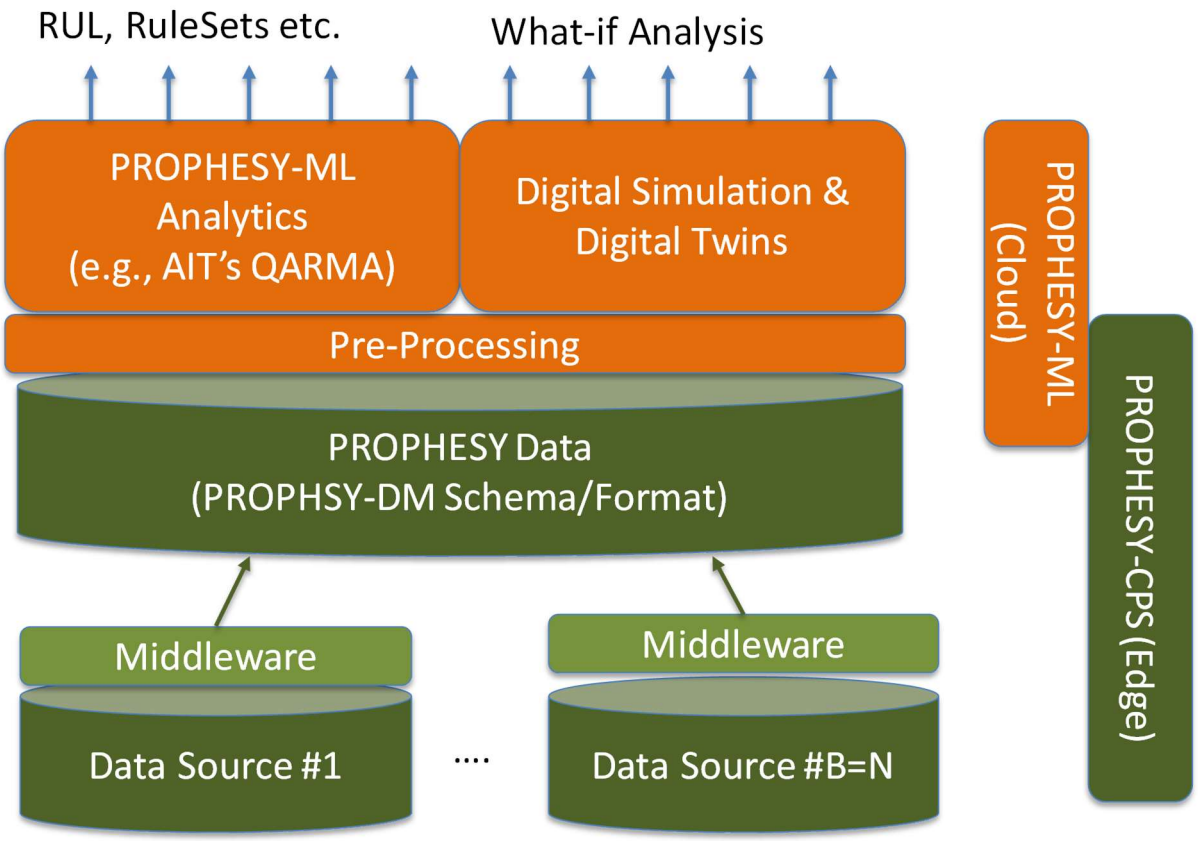


Figure 2: Digital Modelling for Data Representation in PROPHESY

2.1.2 Data Sources and Maintenance Systems Interoperability

Adherence to a given digital model can also facilitate interoperability despite the different formats of the heterogeneous data sources that comprise a maintenance application. The concept has been presented in Figure 2 and repeated in the right hand side in Figure 3, which depicts how different middleware connectors to data sources (e.g., such as sensors, business information systems (e.g., ERP (Enterprise Resource Planning), MES (Manufacturing Execution Systems) and CMMS (Computerized Maintenance Management System)) can be used to convert data stemming from these formats to a common, agreed, standards-based digital model. The latter can be conveniently called PROPHESY-DM (i.e. PROPHESY Data Model). However, the left hand side of Figure 3 illustrates an alternative loosely-coupled approach to interoperability, by means of an interoperability registry. This registry does not base interoperability on adherence to a common format, but rather on a registry that contains information about linking entities and attributes of different digital models and formats. This approach provides interoperability through linking attributes of diverse models. It can be used in as an alternative to the PROPHESY-DM approach, or even in conjunction with it i.e. converting some data formats to PROPHESY-DM and linking attributes and data elements of others to PROPHESY-DM elements.

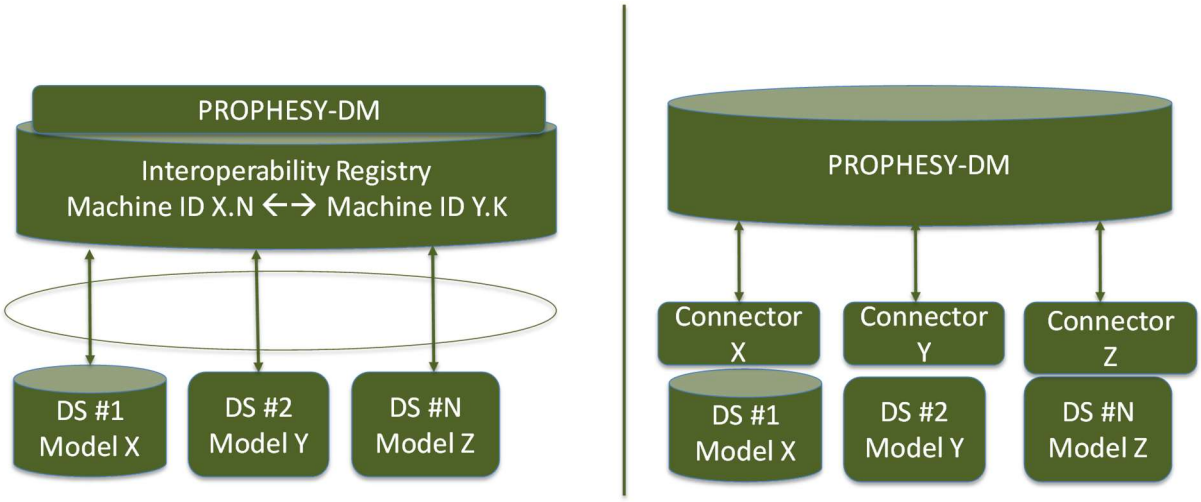


Figure 3: Candidate Interoperability Approaches and Scenarios in PROPHESY

2.1.3 Data Sharing Across Different Systems

Beyond interoperability across heterogeneous data sources and systems that comprise a PROPHESY-PDM deployment, digital modelling can be also used to boost the interoperability across different deployment instances of PROPHESY-PDM, but also across PROPHESY compliant and non-PROPHESY compliant maintenance systems. Figure 4 presents this data sharing and interoperability concept. Datasets can be reused across different deployments as soon as they follow the same (PROPHESY-DM) digital model.

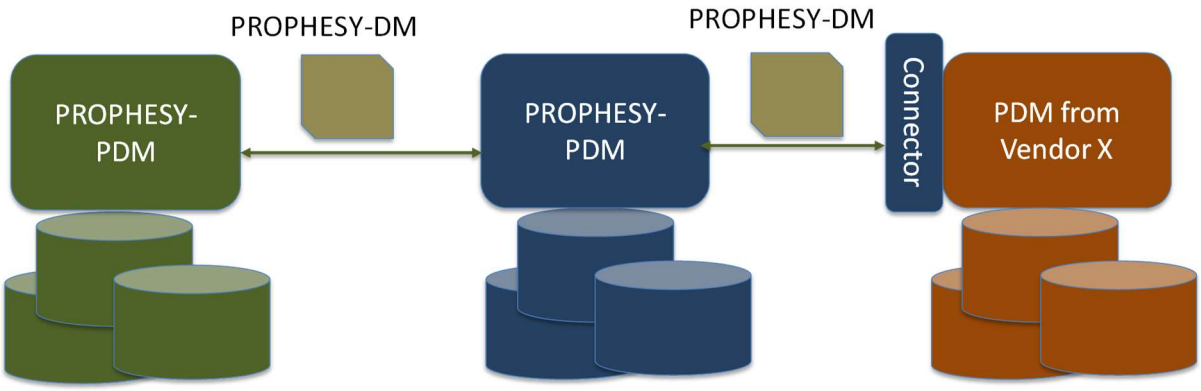


Figure 4: Use of PROPHESY Data Models for Data Sharing Across Different CBM/PDM Systems

2.1.4 PROPHESY Systems Configuration

Another PROPHESY use case that leverages digital models is depicted in Figure 5 and concerns the use of digital models for configuring the various models of a PROPHESY system. To this end, the PROPHESY digital models need to comprise appropriate metadata that enable the configuration of the PROPHESY system. Such metadata can for example concern the physical properties of the data sources (e.g., physical addresses of sensors and CPS devices), the type and semantics of the data that they produce and more. The availability of such metadata

within a configuration tool can enable the automated deployment of data sources and data analytics functions, as well as the support of Plug n' Produce concepts.

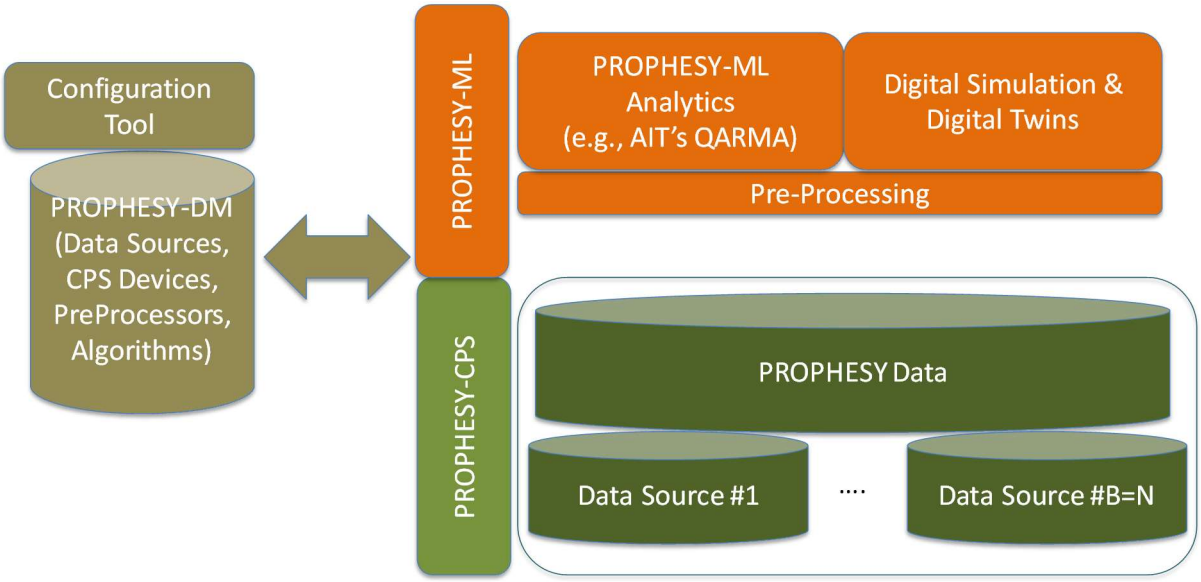


Figure 5: PROPHESY System Configuration Concept

2.2 Digital Modelling Requirements

Based on the above listed use cases, we can also articulate the following requirements for the PROPHESY data models, which include functional, but (mainly) non-functional requirements as well.

2.2.1 Standards Compliance

The PROPHESY digital modelling task should also take into account the adherence to standards-based digital models to the maximum extent possible. This requirement stems from the need to exchange data with third-party systems such as MIMOSA compliant systems.

2.2.2 Extensibility

The PROPHESY digital models must be extensible in terms of the maintenance data sources and the data that they model. The PROPHESY-PDM platform must be seamlessly extensible in terms of maintenance data sources and databases. Likewise, it should be expandable in terms of the supported metadata i.e. providing the means to integrate and leverage additional metadata.

2.2.3 Coverage of PROPHESY Data Sources and Concepts

The PROPHESY Digital Models should be able to support the data sources of the project's use cases, including relevant devices and systems illustrated in respective deliverables, such as deliverable D2.4 that provided an initial description of the PROPHESY use cases in the PHILIPS and JLR factories.

2.2.4 Driven by Reuse of PROPHESY Partners' background Technologies

PROPHESY is an Innovation Action (IA), which emphasizes on the deployment, validation and exploitation of mature solutions rather than researching innovative solutions from scratch. As such the project should leverage readily available digital modelling solutions such as solutions developed in the background projects of the partners. The latter solutions can be used to accelerate the project's developments and improve overall value for money for the partners' developments.

2.2.5 Transformation Capability

The PROPHESY Digital Models, must be flexible transformable to other formats, as part of their sharing across different system and their pre-processing as part of analytics tasks.

2.2.6 Availability of Tools

The PROPHESY digital models should be easily amenable by visual tools in order to facilitate visual deployment and configuration tasks. In this respect, formats for which processing tools exists are preferable from other data formats.

2.2.7 High Performance and Low Latency implementation

The PROPHESY Digital Models should support high performance operations in terms of analysing maintenance data.

2.2.8 Support for Service Oriented Architecture and Interfaces

The PROPHESY Digital Models middleware functionalities (i.e. transformation and integration functions) should provide service oriented interfaces that facilitate their integration with other IT and CPS systems as part of Service Oriented Architectures (SOA).

3 Review of Digital Models for Industry4.0, Industrial Automation and Predictive Maintenance

3.1 Data Models for Industrial Automation

3.1.1 Relevant Standards

There is a long list of standards that provide digital models for representing plant information and industrial automation processes. A detailed description of these standards is beyond the scope of this document, as interested readers can refer to relevant literature (e.g., [1] [2]). Moreover, some of the most prominent of these standards have been presented as part of deliverable D2.1 of the project. In the following table we provide a list of relevant standards, along with information about their relevance for PROPHESY:

Standard	Description & Use	Relevance for PROPHESY
IEC 61512 BatchML	XML based implementation of the ANSI/ISA-88 Batch Control family of standards.	Suitable for the modelling of ISA-88 compliant systems, which is not the case for PROPHESY.
IEC 62769 (FDI)	Includes an information model that represents automation systems' topologies, including field devices and the communication networks that interconnect them.	Suitable for modelling field information (devices, networks), but without provisions for data analytics like in PROPHESY WP4.
IEC 62264 B2MML	XML based specification and implementation of the ANSI/ISA-95 family of standards.	Very good choice for modelling interactions across entities within MES and ERP systems and their involvement in automation operations. However, it does not support modelling of analytics operations.
ISO 15926 XMplant	Provides support for digital modelling of plant information, based on the ISO 15926 specification. It covers the structure, the geometry and 3D models about a plant.	It is a good choice for modelling the static elements and behaviours of a plant. However, in order to be used in PROPHESY, it needs to be extended with additional data/metadata for maintenance information.
IEC 62424 CAEX	Supports XML-based representation of plant information, including all the components of a plant in an	CAEX can cover the modelling of the plant elements, but is in appropriate for modelling maintenance-related

	hierarchical structure. It adopts an object-oriented philosophy.	information such as sensor-based datasets.
IEC 62714 AutomationML	Provides an XML-based format for the description of engineering and automation processes in a plan. It is based on three other standards, namely CAEX (for plant information), COLLADA (for geometry and kinematics) and PLCopen (for control applications).	AutomationML can provide a standards-based way for describing automation elements and operation in PROPHESY. Nevertheless, it needs to be complemented with other models that will cover the maintenance elements of a PROPHESY deployment, including analytics and streaming data.
OPC UA's Data Model	Defines a generic object oriented Data Model (DM), which focuses on devices representation.	OPC UA DM is a good choice when OPC UA is used for connecting to the field. Nevertheless, it is inadequate to model/support other elements of a PROPHESY maintenance deployment (e.g., maintenance related datasets).
MTConnect	Provides an XML-based format for exchanging data between the shop-floor and IT applications, including data about devices, topologies and components characteristics.	It's also a standard that covers plant information (devices/topologies), without however supporting fine-grained information about maintenance applications.

Table 1: Standards-based Data/Information Models for Industrial Automation

As evident from the above-listed table, industrial automation standards come with information models that enable the modelling of plant topologies, devices, data sources and communication information about a plant. However, they do not provide the means for supporting two key requirements for PROPHESY, namely: (i) the modelling of the maintenance context (e.g., machinery information, failure modes, service information) and (ii) the modelling of analytics processes and tasks, which will be used in PROPHESY in order to compute predictive maintenance parameters (e.g. RUL) and provide related recommendations. As such these information models can be used as part of the PROPHESY digital modelling infrastructure in order to support automation related modelling, but cannot support the full range of PROPHESY requirements.

3.1.2 Review of Relevant Projects

In following paragraphs, we also review two the digital models that have been developed in two of the projects where the partners participate, namely H2020 PERFORM and H2020 FAR-EDGE. Similarly, to the standards of the previous paragraph, these projects are not focused on enterprise maintenance, but rather on industrial automation. As such their models can

only cover the automation aspects of the PROPHESY platform (e.g., digital twins and field interaction use cases that entail automation processes).

3.1.2.1 Data Modelling in H2020 PERFORM: PERFoRMML

The H2020 PERFORM project is devoted to the development and validation of a plug n' produce infrastructure. The latter is destined to enable the seamless connectivity and integration between both smart and legacy production systems. To this end, the project has design a common data model, along with two standards-based interfaces. The development of the model is driven by the needs of four plug n' play use cases.

Following a comprehensive review and evaluation of various data models, the PERFORM consortium has selected AutomationML as the base for building its own common data model, which is conveniently called PERFoRMML. PERFoRMML is represented as a class diagram, which makes provisions for modelling/representing the following entities and their relationships:

- **Machinery and Control Systems**, which provide the means for modelling the topology, data types and interactions of production systems at physical machinery level. The attributes of these entities enable capturing and modelling of parameters for configurations and skills, as well as for shop-floor data to be extracted from various sources such as PLCs and databases. In particular, the following sub-entities are also modelled through proper subclasses:
    - **Skills** (e.g. pick, place, move, weld) refer to abilities, functions or tasks performed by shop-floor elements. They may possess and certain values that are relevant to be extracted (e.g., cycle time, energy consumption, sensor data).
    - **Configurations** provide a high-level description of a possible configuration to execute a given skill, according to a set of specified parameters.
    - **Products**, which correspond to abstractions of given product variants, along with their core-defining characteristics to enable a process-oriented description of the product.
    - **Processes**, which present a description of the ordered steps required for the production of an associated product.
    - **Connectors**, which encapsulate and abstract the information required in order to communicate with components in the shop-floor. Their abstraction property enables them to model and support communications regardless of the actual communication protocols (e.g., OPC-UA, MQTT) used.
    - **Events**, which model certain occurrences in production that may require the attention of the system or its users.
- **Data Backbone entities**, which models the elements that are necessary for interactions with the tools connecting to the PERFORM middleware. Using this middleware these entities can acquire data and information from the lower-level and act based on it. Such entities include:

- **System**, which is an entity representing entire production systems and therefore comprises systems information in terms of topology, products and possible simulations.
- **Simulation Results**, which support the representation of some simulation outcome (usually a KPI (Key Performance Indicator) in-line with the PERFORM's digital twins requirements.
- **Schedules**, which model the allocation of the (end-to-end) steps that need to be executed in order for the production of certain product.

For each of the two sets of entities (machinery & control, data backbone) the project specified standard based interfaces for accessing instance data of the various entities. These interfaces form the basis for an API (Application Programming Interface) as well. As already outlined, PERFORM provides some useful concepts that can be used in PROPHESY, yet it is focused on automation concepts (e.g., schedules and production processes) rather than predictive maintenance concepts (e.g., data analytics, faults, failure mode, FMECA (Failure Mode, Effects, and Criticality Analysis) concepts).

3.1.2.2 Digital Modelling in FAR-EDGE

The main goal of H2020 FAR-EDGE is to provide and validate a novel industrial automation platform that leverages the edge computing paradigm, along with distributed ledger technologies for secure state synchronization of edge industrial systems in a plant or across the supply chain. As part of its platform design and implementation, FAR-EDGE has also developed a range of digital models for the three main functional domains of its platform, namely automation, simulation and Distributed Data Analytics (DDA). In following paragraphs, we concentrate on the DDA modelling part, which is the most relevant to PROPHESY, as it enables the representation of streaming data sources in the factory and the analysis of their data by means of data analytics algorithms. Hence, although not explicitly designed for maintenance, the digital models that support the DDA part of FAR-EDGE have a clear functional relevance to the PROPHESY project.

In a nutshell, the proprietary FAR-EDGE data models for DDA are used for configuring distributed data analytics functionalities, model factory data and metadata, along with the analytics functions and workflows that process them. More information about the concepts and the entities that they can model at the cyber/digital part of manufacturing systems are provided in the following paragraphs.

3.1.2.2.1 Factory Data and Metadata

The FAR-EDGE digital models support the representation of factory data and metadata based on the following entities:

- **Data Source Definition (DSD)**: This defines the properties of a data source in the shopfloor, such as a data stream from a sensor or an automation device.
- **Data Interface Specification (DI)**: The DI is associated with a data source and provides the information need to connect to it and access its data, including details like network protocol, port, network address and more.
- **Data Kind (DK)**: This specifies the semantics of the data of the data source, which provides flexibility in modelling different types of data. The DK is an XML specification and hence it can be used to define virtually any type of data in an open and extensible way.
- **Data Source Manifest (DSM)**: A DSM specifies a specific instance of a data source in-line with its DSD, DI and DK specifications. Multiple manifests (i.e. DSMs) are therefore used to represent the data sources that are available in the factory in the scope of the FAR-EDGE automation platform.
- **Data Consumer Manifest (DCM)**: This models an instance of a data consumer i.e. any application that accesses a data sources.
- **Data Channel Descriptor (DCD)**: A DCD models the associated between an instance of a consumer and an instance of a data source. It is useful to keep track of the established connections and associations between data sources and data consumers.
- **LiveDataSet**: This entity models and represents the actual dataset that stem from an instance of a data source that is represented through a DSM. Hence, it references a DSM, which drives the specification of the types of the attributes of the LiveDataSet in-line with the DK. A LiveDataSet is associated with a timestamp and keeps track of the location of the data source in case it is associated with a mobile (rather than a stationary) edge node. Hence, it has a location attribute as well. In principle the data source comprises a set of name-value pairs, which adhere to different data types in-line with the DK of the DSM.
- **Edge Gateway**: This entity models an edge gateway of a FAR-EDGE edge computing deployment. In the scope of a FAR-EDGE deployment, data sources are associated with an edge gateway. This usually implies not only a logical association, but a physical association as well i.e. an edge gateway is deployed at a station and manages data sources in close physical proximity to the station.

Based on the above entities it is possible to represent the different data sources of a digital shopfloor in a modular, dynamic and extensible way. This is based on a repository (i.e. registry) of data sources and their manifests, which keeps track of the various data sources that register to it. The FAR-EDGE platform includes such a registry, which provides dynamicity in creating, registering and using data sources in the industrial plant.

3.1.2.2.2 Factory Data Analytics Metadata

In order to facilitate the management and configuration of analytics functions and workflows over the various data sources, the FAR-EDGE digital models specify a number of analytics-related entities. In particular:

- **Analytics Processor Definition (APD)**: This specifies a processing function to be applied on one or more data sources. In the scope of FAR-EDGE, three processing functions are defined, including functions that pre-process that data of a data source (i.e. Pre-Processors), functions that store the outcomes of the processing (i.e. Store Processors) and functions that analyse the data from the data sources (i.e. Analytics Processors). These three types of processors can be combined in various configurations over the data sources in order to define different analytics workflows.
- **Analytics Processor Manifest (APM)**: This represents an instance of a processors that is defined through the APD. The instance specifies the type of processors and its actual logic through linking to a programming function. In the case of FAR-EDGE, the latter is a class/programme implemented in the Java language.
- **Analytics orchestrator Manifest (AM)**: An AM represents an entire analytics workflow. It defines a combination of analytics processor instances (i.e. of APMs) that implements a distributed data analytics task. The latter is likely to span multiple edge gateways and to operate over their data sources.

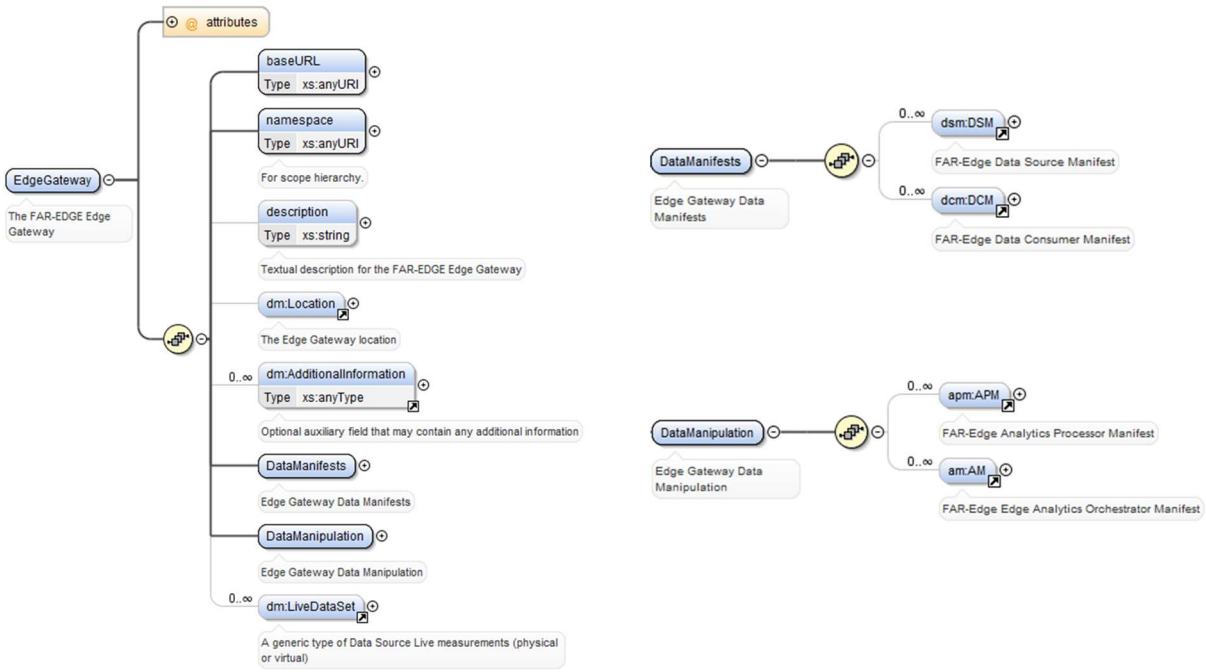


Figure 6: Snapshot of the FAR-EDGE Digital Models Structure

The FAR-EDGE Digital Models for distributed data analytics follow a hierarchical structure, which defines the different relationships between the various entities. For example, an edge gateway comprises multiple data source manifests. Each one of the latter is associated with a data source definition. Likewise, LiveDataSets are associated with instances of data sources i.e. data sources manifests. As an example, Figure 6 illustrates a snapshot of the FAR-EDGE digital models structure, which shows the association of each edge gateway with data source manifests and data analytics manifests. A more detailed presentation of the hierarchical structure of our data models is beyond the scope of this chapter. Interested readers can

consult directly our XML schemas, which are part of our open source implementation of the FAR-EDGE digital models repository that is also an integral part of the FAR-EDGE platform.

3.2 Data Models for Condition Based Monitoring and Maintenance

3.2.1 Relevant Standards

The MIMOSA association is producing a set of standards for condition monitoring and maintenance, which can be used for standardizing information representation and access as part of predictive maintenance applications. In particular, MIMOSA is a non-profit trade association dedicated to developing and encouraging the adoption of open information standards for Operations and Maintenance in area such as manufacturing, fleet, and facility management. In the sequel we review the scope and rationale of some MIMOSA standards that are relevant to PROPHESY.

3.2.1.1 MIMOSA OSA-EAI and MIMOSA Information Architecture

The MIMOSA Open Systems Architecture for Enterprise Application Integration (OSA-EAI) specifications are destined to address one of the most common problems of the development of maintenance systems, which is the fact that information and data are fragmented across different systems such as engineering, maintenance, operations and reliability related systems. As outlined, this is a requirement for the PROPHESY platform as well, which seeks to integrated data residing in various disaggregated sources and databases. However, the specification of a standards-based information model for maintenance and operations data serves most of the PROPHESY requirements and use case listed in Section 2, since it:

- Facilitates a common and unified understanding of business requirements, which boosts communication and collaboration across team members. The latter can be facilitated by the establishment and use of a common naming scheme for the maintenance-related data.
- It provides a foundation for designing databases and datastores where maintenance information will be persisted.
- It boosts data re-use, data sharing and data repurposing across different maintenance systems operated in different factories and across the supply chain.
- Accelerates development and maintenance, while reducing costs and leading to an overall better ROI for any maintenance application that integrates data from different sources/systems.
- Fosters seamless communication across different systems and applications in the supply chain.
- Helps data analytics teams in their data pre-processing and analytics efforts, as they have to deal with a single (rather than multiple) digital model and format.

The OSA-EAI specifications are based on the representation of the information objects required for a business applications based on information model that adheres to OpenO&M

standards. Such a model contains the things/entities of importance for a business domain and an organization, along with their relationships. It provides a basis for designing a database, that will store relevant data, without imposing however a specific database management technology. The MIMOSA information model complies with ISO standards such as:

- **ISO 13374-1**: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 1: General guidelines and
- **ISO 13374-2**: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 2: Data processing.

The MIMISA ISO 13374 compliant information architecture, builds upon a Common Conceptual Object Model (CCOM), which is converted into a Relational Implementation Model i.e. the Common Relational Information Model (CRIS). CRIS enables the implementation of the MIMISA information model in Relational Database Management Systems (RDBMS). The model contains information about reliability and maintenance, including condition event data, condition measurement data (i.e. scalar data, dynamic data (e.g., sensor data such as vibration and sound)), information for assessing the health of assets and more.

On top of the MIMOSA CRIS, the MIMOSA-EAI provides a set of service oriented interfaces for accessing the databases through different IT systems and applications. The concept is illustrated in Figure 7.

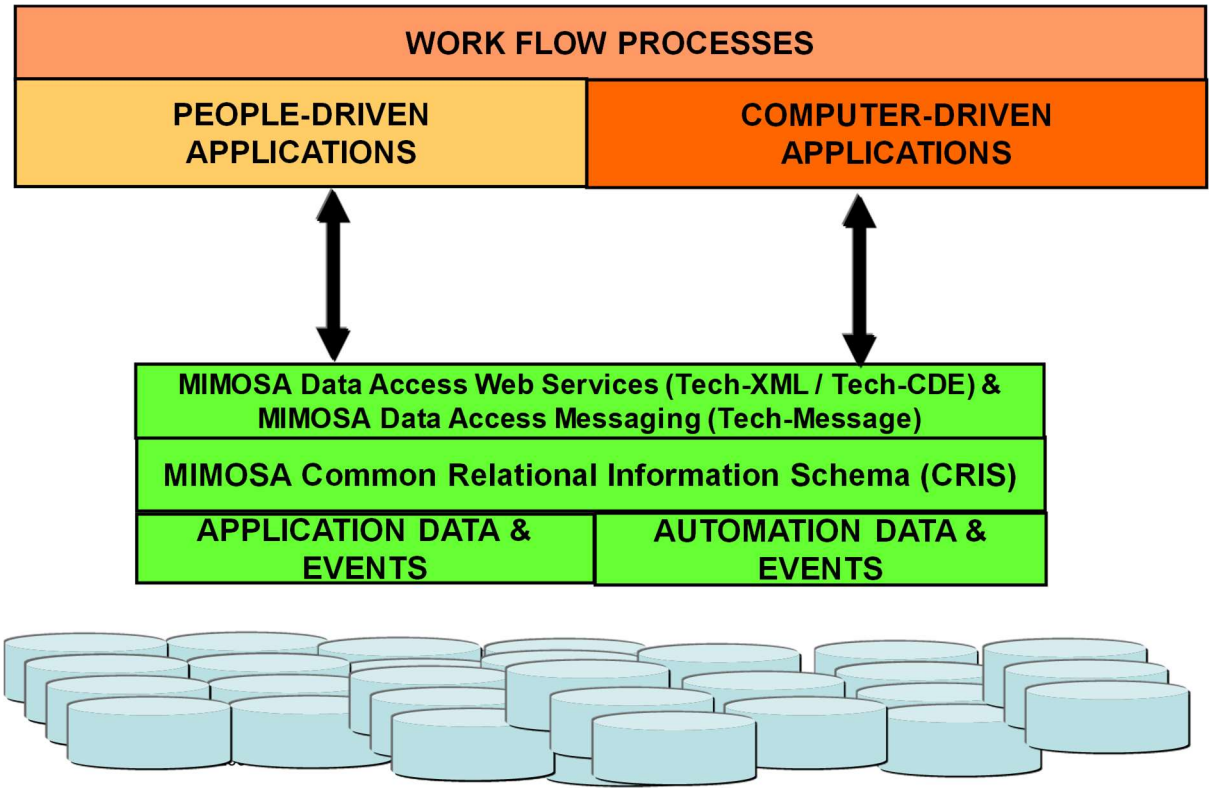


Figure 7: MIMOSA OSA-EAI Concept – Role of the Common Relational Information Schema (CRIS)

3.2.1.2 MIMOSA OSA-CBM

Similarly, to OSA-EAI the OSA-CBM specification is a standard architecture for moving information in a condition-based maintenance system. It leverages the information model listed earlier and offers four interfaces for accessing relevant information:

- **A Synchronous interface**, where Information is returned by Request method.
- **An Asynchronous interface**, Information is returned as available via an Asynchronous data path for requested information.
- **A Data Service interface**, including services that define one-way input data devices.
- **A DataEvent Server Interface**, which is an interface mechanism for handling an individual DataEvent in a simplified interface.

As already outlined, it leverages the OSA-EAI CRIS information in order to access six main categories of information, including dynamic data, configuration data, explanation data, control data, application data and error data. Each of these categories of information are individually addressable through the previously presented interfaces.

An OSA-CBM Software Development Kit is available for the .NET framework, including code and schemas about the data and the interfaces for accessing the information.

3.2.1.3 MIMOSA Registry Management

The MIMOSA Information model (e.g., CCOM/CRIS) can serve as a basis for implementing registries of condition monitoring and maintenance data entities, such as registries of enterprises, sites, physical assets and various other assets' resources (e.g., parts, consumables, tools, labour). These include metadata that are needed to develop maintenance applications and as such are very relevant to PROPHESY. In essence they provide the maintenance-related information that is missing from all presented information models about industrial automation processes.

3.2.1.4 Use of MIMOSA in PROPHESY

Experience from attempts to use MIMOSA in earlier projects of the partners (including the MANTIS project) has revealed that it is a heavyweight model with many modelling concepts and artefacts. The latter introduce complexity in every effort to customize MIMOSA models for condition monitoring and maintenance applications, especially in cases where other aspects of MIMOSA (e.g., reliability) are not of interest. In particular, the CRIS schema includes too many relational tables which makes relevant modelling efforts complex and introduce overheads at the design, implementation and operations phases.

Nevertheless, the MIMOSA models comprise all essential modelling constructs, including the required metadata for maintenance applications. Hence, PROPHESY can benefit from these modelling constructs in the form of a lightweight MIMOSA representation that can be integrated in PROPHESY-DM infrastructure.

3.2.2 Review of Relevant Projects (H2020 MANTIS)

MANTIS takes advantage of its own digital models in order to boost interoperability. These models are provided in an abstract, technology agnostic form, which can be instantiated and customized in the scope of a concrete MANTIS implementation. This is evident in the MANTIS application integration model, which shows how a conceptual integration approach is transformed in a specific maintenance system implementation within a concrete technical environment Figure 8.

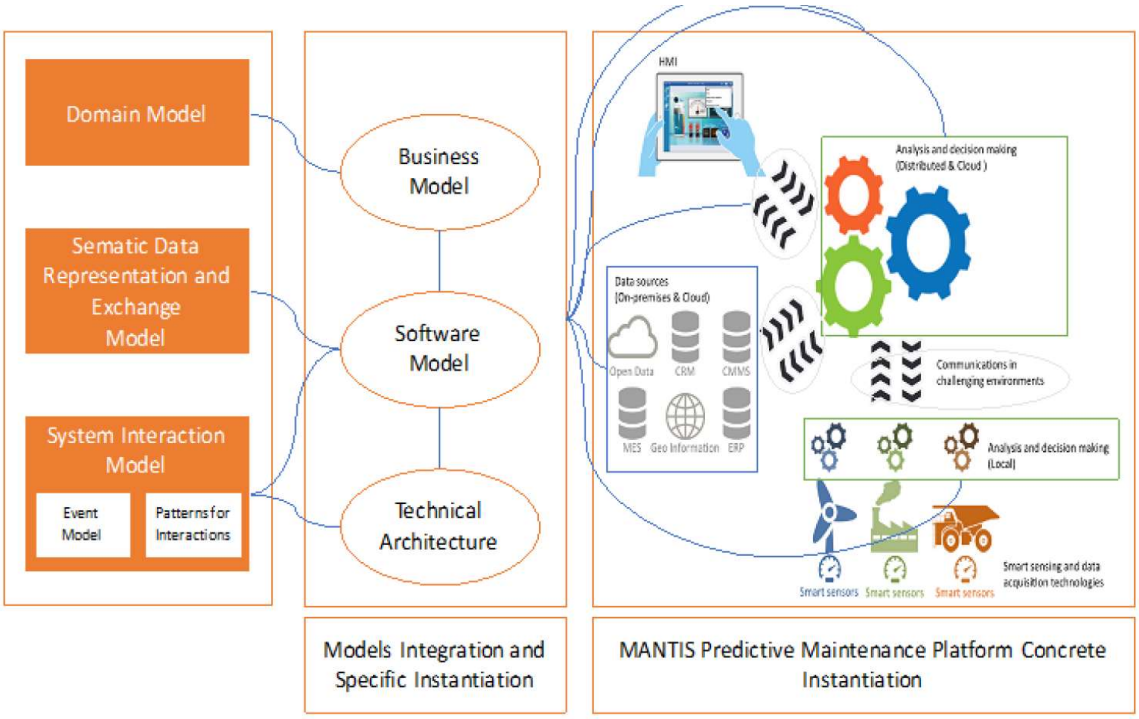


Figure 8: MANTIS Application Integration Framework

The framework for concrete MANTIS implementations (i.e. the technical framework) is illustrated in Figure 9. These models illustrates also the MANTIS interoperability perspective. It is based on various cyber-physical systems (CPSs), core services and related execution engines, processes, data analytics processors, tools and applications that are aimed to support maintenance activities. A Service Oriented Architecture (SOA) approach is followed, to integrate services from various software systems and/or platforms.

The technical framework is based on a 4-tier model that covers data representation and exchange, system interactions (event models and patterns for interaction), data transformation and translation and services and functionalities definition, as well as, physical entities virtualization. The SOA approach is used to couple the services in the various tiers. The services include mechanisms for discovering, composing and orchestrating the services. Moreover, a MIMOSA data storage infrastructure is used to facilitate the design and implementation of maintenance applications within the business tier. This infrastructure

comprises events, metadata and the result of the data analysis processes based on appropriate translators and transformations that ensure adherence to MIMOSA formats.

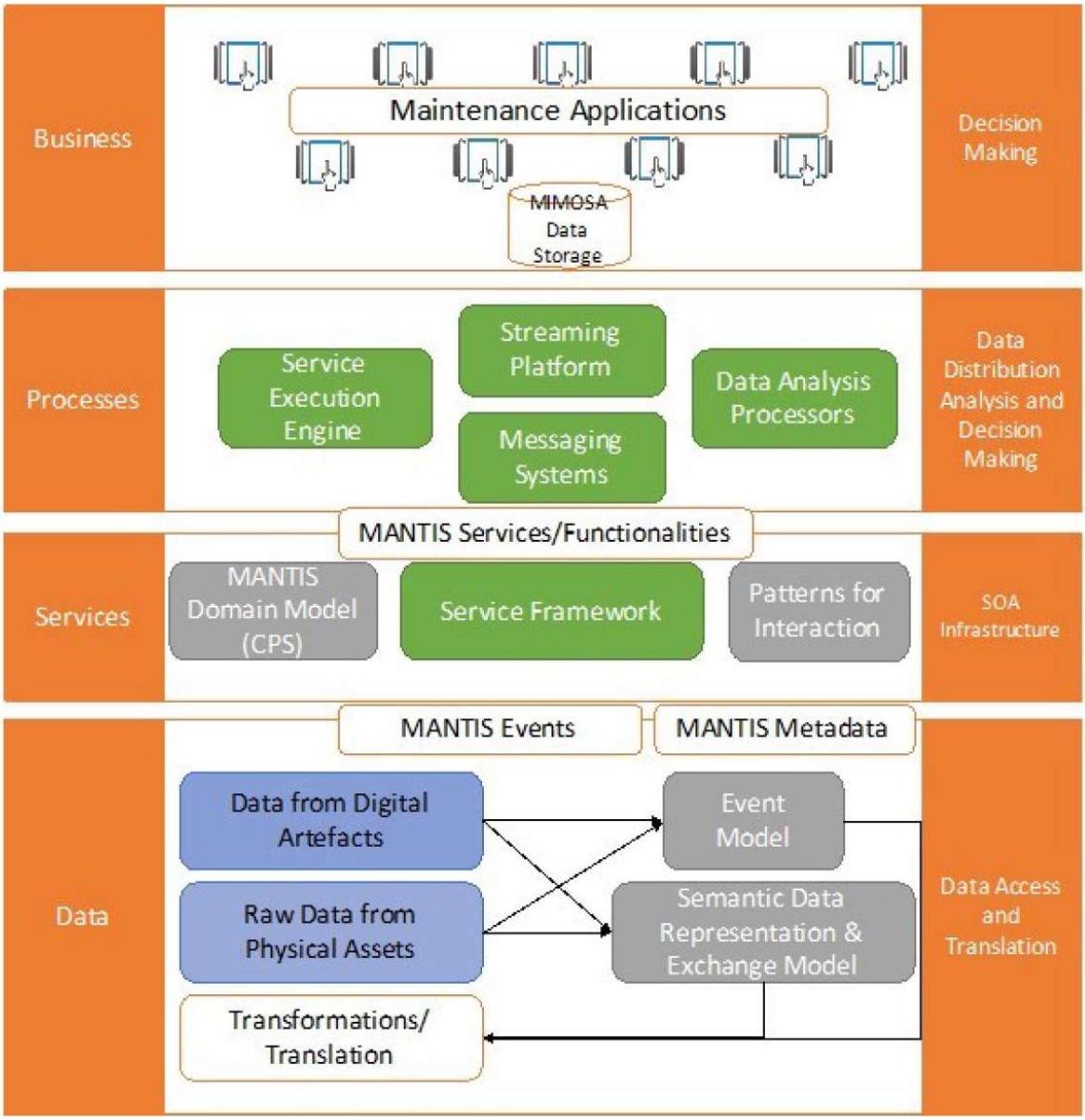
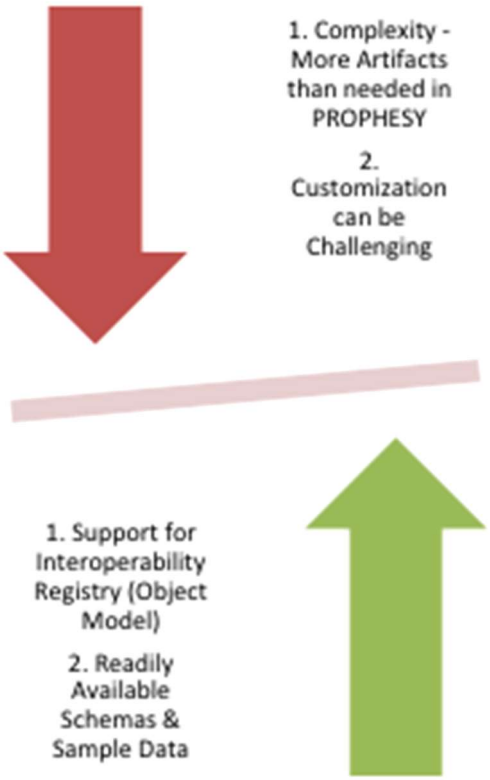


Figure 9: MANTIS Technical Integration Model

4 Synthesis – PROPHESY Data Models Specification

4.1 Design Decision and Trade-offs

As already outlined in Section 2, the PROPHESY data models should support the storage and management of data and metadata that are essential for configuring and executing the PROPHESY applications, including legacy systems and applications that are integrated in the PROPHESY platform (e.g., the MMS data collection system for the MAG machines). In this effort, we were confronted with the following design dilemma:



- **Option #1 (Complete, yet heavyweight approach):** Developing a MIMOSA-based or MIMOSA-like schema that will comprise the vast majority of concepts that are used in PROPHESY data analytics and visualization applications. Such an approach would provide completeness in terms of the PROPHESY data representation, yet it would require transforming data from other applications (and their formats) to the PROPHESY data modelling format. Hence, despite its completeness, a significant number of middleware transformation would be required for each new data source and data format to be handled.
- **Option #2 (Linking, i.e. lightweight schema linked to other readily available schemas)**: Providing a lightweight schema that captures a core set of PROPHESY-related entities and linking it to other schemas. This approach strives to exploit available data presentations and datasets, through linking them around the PROPHESY data models. In particular, the approach is based on the linking of information from different repositories for a given object (e.g., sensors, machine, tool). Hence, it provides flexibility in linking more repositories as part of the extensibility of the PROPHESY infrastructure. Any database or repository that contains maintenance related information (i.e. data or metadata) about a PROPHESY application can be linked to other schemas at the level of an entity. This approach can be implemented based on a Common Interoperability Registry approach.

Taking into account: (i) The requirements listed in Section 2; (ii) On-going developments in the PROPHESY-CPS platform specifications and (iii) Work in progress associated with the implementation of the PROPHESY service bricks, we opted for the second of the above options as a means of ensuring flexibility and extensibility in leveraging different datasets based on minimal effort (i.e. based on linking rather than middleware implementation).

Based on the specification and implementation of an object identification registry, we therefore aimed at achieving the following goals:

- Supporting both static and dynamic data sources in order to cover diverse factory monitoring and predictive maintenance scenarios. In this context, static data sources can have known structure (i.e. schema).
- Supporting storage of observed information for a great variety of different data sources, through linking the observations with the objects that they concern in the PROPHESY data models.
- Supporting the use of third party databases as a means of enriching data observations with deployment (scenario-specific) semantics. This can be done through linking entities in the PROPHESY data models with metadata about these entities in other (linked) repositories.

Overall, the PROPHESY Digital Models solution is based on an Interoperability Registry approach. Specifically, the data models comprise a core database of PROPHESY data and metadata, as well as its linking to other repositories that contain relevant information through the registry.

4.2 Elements of the PROPHESY Data Modelling Infrastructure

An outline of the PROPHESY Digital Modelling solution is provided in Figure 10 and comprises the following main elements:

- **The PROPHESY Digital Models database**, which contains the main structure of the maintenance data that are used in PROPHESY. The models define the structure of the observations that will be derived by sensors and other modules of a PROPHESY deployment.
- **A lightweight MIMOSA-based database**, which provides the basic maintenance metadata that are used within the PROPHESY applications.
- **The Common Interoperability Registry,** which provides the infrastructure for linking objects and their information that reside in different databases. This linking enables the enrichment of the PROPHESY datasets based on additional data and metadata residing in the linked databases.

Based on these elements, PROPHESY applications will be provided with the means to access consolidated datasets from all linked databases via the CIR. In particular, applications will

consumer PROPHESY-related data for given objects. The CIR will provide the means for accessing not only data for this object in the PROPHESY digital database, but also metadata and data from linked databases. In this way, they will be able to access enhanced datasets along with proper metadata that describe the context of the objects at hand.

Note also that Figure 10 depicts a Data Management console as a core element of the infrastructure. This console is aimed at enabling the configuration of the CIR and the databases of the infrastructure through a single point access.

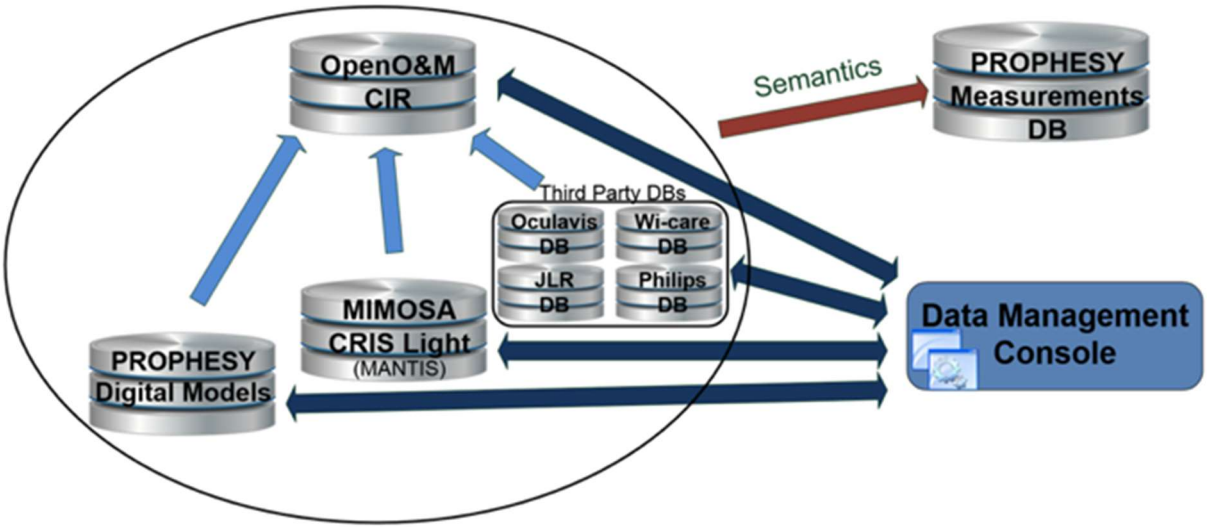


Figure 10: PROPHESY Data Models and Interoperability Registry

4.3 PROPHESY Digital Models Implementation Infrastructure

In order to rapidly bootstrap the implementation of the digital models infrastructure the project will take advantage of some of the background developments listed in Section 3, notably developments of the project partners in other projects (Table 2). In particular:

- **The FAR-EDGE Digital Models infrastructure** will be customized to the requirements for the modelling of PROPHESY data observations. FAR-EDGE provides a readily available infrastructure for modelling streaming data sources, including information about connecting to them and modelling of their semantics and live data. In PROPHESY, the FAR-EDGE schemas (e.g., XML schema elements like DSD, DI and DK) are used to represent PROPHESY streaming data.
- **The MANTIS MIMOSA/CRIS database** will be used a basis for modelling and storing metadata about maintenance related objects, including for example information about the vendors, location and use of the various objects. The linking of these metadata with the data of the objects will enable PROPHESY applications to access complete maintenance information for their analytics and visualization needs.
- **The Open O&M CIR,** will be used as an infrastructure for linking information about objects, despite the fact that they will reside in different repositories.

Using the above listed elements, a PROPHESY application will be able to consult and access (through the CIR) information about the full context and observations that related to an object, regardless of the repository they reside. Likewise, the infrastructure will enable the flexible extension of the PROPHESY digital models' infrastructure will information (data/metadata) stemming for additional repositories and databases. Nevertheless, note that this will require each new repository that extends the data and metadata of the PROPHESY digital models infrastructures to be linked to objects of the PROPHESY database at the time of their deployment.

Component	Implementation Technology	Comments & Remarks
PROPHESY Digital Models	FAR-EDGE Digital Models	FAR-EDGE Schemes for Dynamic Data Sources (e.g., DSD) customized for PROPHESY
Maintenance Metadata	MANTIS MIMOSA Infrastructure (lightweight CRIS Schema)	The Infrastructure will be used as is, without ruling out the possibility of enhancing it with more metadata
Interoperability Registry	Common Interoperability Registry from Open O&M	The CIR specification will be customized to support linking of PROPHESY related datasets

Table 2: Implementation Technologies for the PROPHESY Digital Models Infrastructure (including the Interoperability registry)

4.4 PROPHESY Models for Data Routing & Data Exchange Functionalities

4.4.1 Digital Models Schema

As mentioned above PROPHESY will use a customized version of FAR-EDGE Digital models capable to support the measurements streaming from the monitored machines along with a data routing and configuration methodology. In the following sections we describe this customized data model and we provide the equivalent XML schemata in Annex A.2 below.

4.4.1.1 DM root entity

The root element of the PROPHESY Digital Models (PROPHESY-DM) is the "ProphesyDM" and is depicted in Figure 11 below. The XSD schema of the "ProphesyDM" is provided in Table 4 of Annex A.2 below. The ProphesyDM schema includes 7 additional schemata which names, namespaces and schema filename are listed in Table 3 below

Name	Namespace	Prefix Used	Root Element	Schema File Name
Data Kind	eu:prophesy:drpp:dk	dk:	DK	DataKind.xsd (can be found in Table 7)
Data Interface	eu:prophesy:drpp:di	di:	DI	DataInterface.xsd (can be found in Table 8)
Data Source Definition	eu:prophesy:drpp:dsd	dsd:	DSD	DataSourceDefinition.xsd (can be found in Table 6)

Data Source Manifest	eu:prophesy:drpp:dsm	dsm:	DSM	DataSourceManifest.xsd (can be found in Table 5)
Processor Definition	eu:prophesy:pd	pd:	PD	ProcessorDefinition.xsd (can be found in Table 11)
Processor Orchestrator	eu:prophesy:processor: orchestrator	po:	PO	ProcessorOrchestrator.xsd (can be found in Table 9)
Processor Manifest	eu:prophesy:pm	pm:	PM	ProcessorManifest.xsd (can be found in Table 10)

Table 3: ProphesyDM imported schemata

As depicted in Figure 11 below the "ProphesyDM" has:

- **id**: A required ID which identifies the PROPHESY Digital Model instance.
- **name**: An optional human recognisable name of the PROPHESY Digital Model instance.
- **description**: which provide a description of the PROPHESY Digital Model instance.
- **AdittionalInformation** (unlimited list): which provides an optional unlimited auxiliary field that may contain any additional information and it is used for further extensions. More specifically, it serves as the root of the type definition hierarchy for any schema and it has the unique characteristic that it can function as a complex or a simple type definition, according to context. More details are provided in section 4.4.1.6.2 below.
- **DataDefinitions**: Here the different PROPHESY reusable models are listed. These models need to be specified in a global level (i.e., PdM) prior of their instantiation and usage in a local level (I.e. CPS) and are modelling data definitions, interfaces and processors. More details are provided in section 4.4.1.2 below
- **DataManipulation**: Here the different reusable models capable of manipulating data are listed. These models are instansiation of models specified in the Data Definition above and can be used globally (i.e., PdM) or locally (i.e., CPS). More details are provided in section 4.4.1.3 below.
- **EdgeGateway** (unlimited list): which is the most important entity of the Digital Model and maps the activity of the PROPHESY CPS layer. This entity is analysed in the section 4.4.1.4 below.
- **LiveDataSet**: the Live Data Set provides a structure capable of hosting the data streams and data sets of PROPHESY. More details are provided in section 4.4.1.5 below.

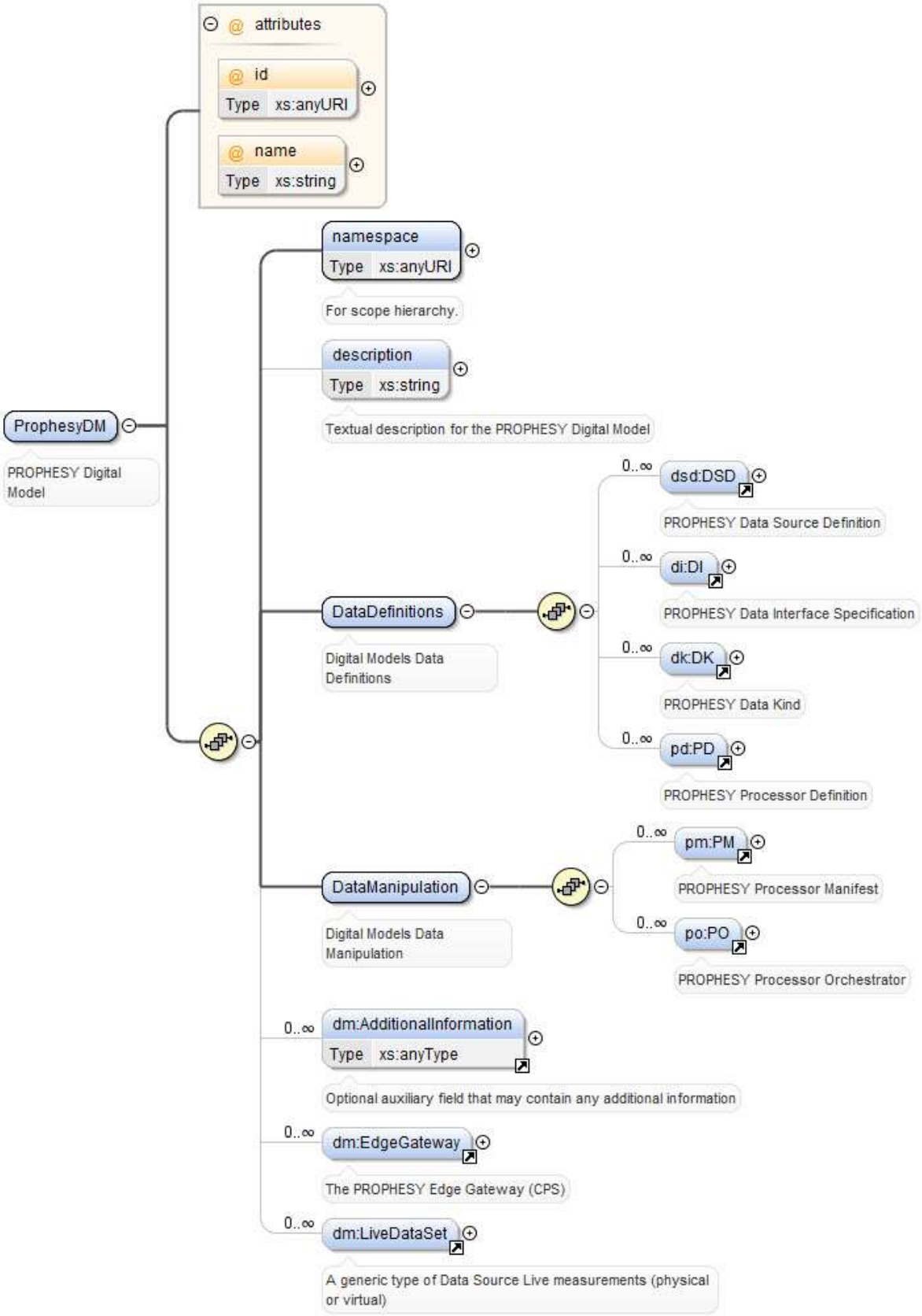


Figure 11: PROPHESY DM root entity

4.4.1.2 Data Definitions

4.4.1.2.1 Data Kind (DK)

This entity specifies the semantics of the data of the data source, which provides flexibility in modelling different types of data. It can be used to define virtually any type of data in an open and extensible way. The root element of the Data Kind entity is the "DK" and is depicted in Figure 12 below. The XSD schema of the "DK" is provided in Table 7 of Annex A.2 below.

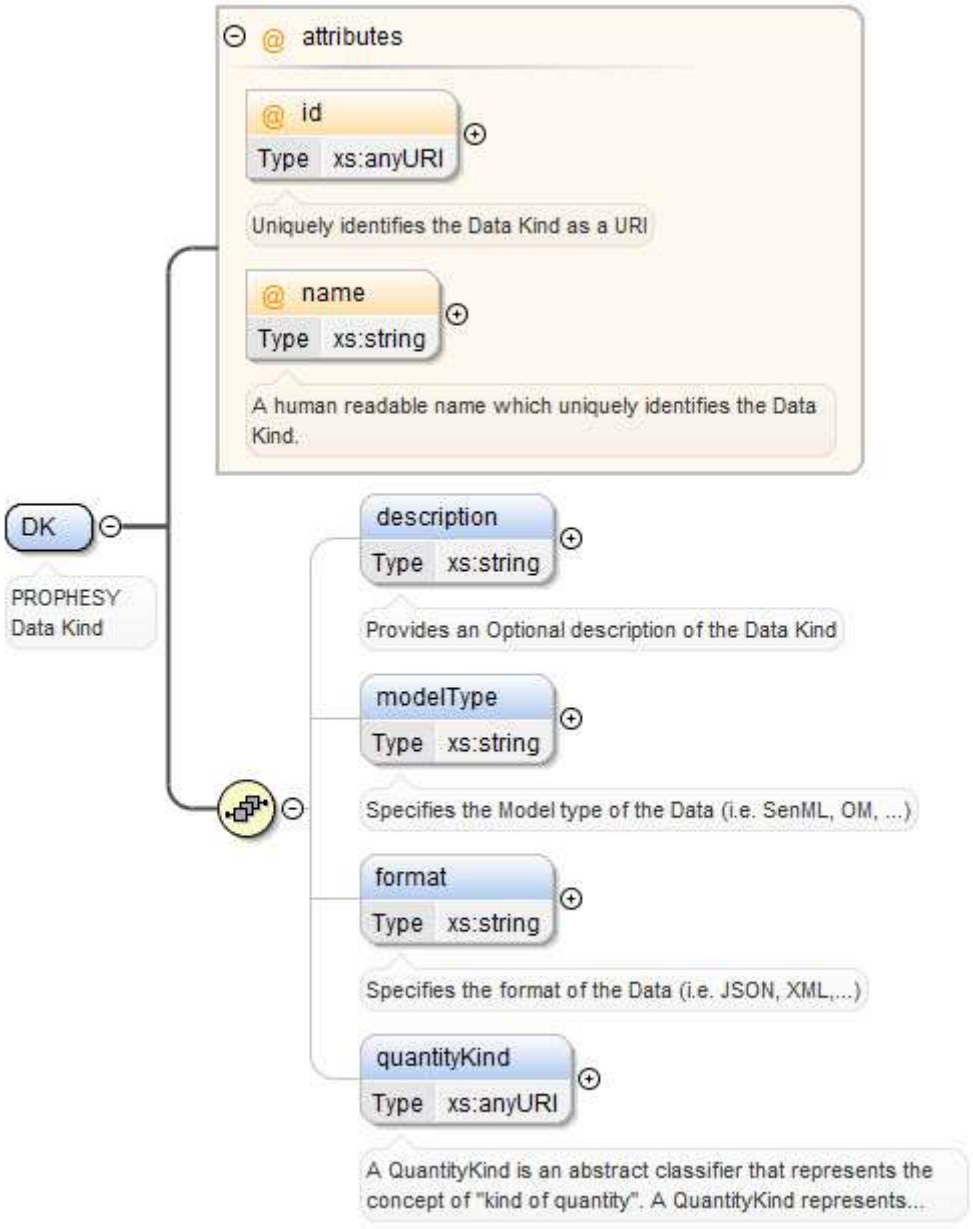


Figure 12: PROPHESY Data Kind entity

As depicted in Figure 12 above the "DK" has:

- **id**: A required ID which uniquely identifies a DataKind within a PROPHESY deployment
- **name**: An optional human-readable name which uniquely identifies the DataKind

- **description**: Provides an optional description of the DataKind
- **modelType**: Specifies the model type of the Data (i.e. SenML, OM, ...)
- **format**: Specifies the format of the Data (i.e. JSON, XML,...)
- **quantityKind**: A QuantityKind is an abstract classifier that represents the concept of "kind of quantity". A QuantityKind represents the essence of a quantity without any numerical value or unit. (e.g. A sensor -sensor1- measures temperature: sensor1 has quantityKind temperature).

4.4.1.2.2 Data Interface (DI)

The DI entity is associated with a data source and provides the information need to connect to it and access its data, including details like network protocol, port, network address and more. The root element of the Data Interface entity is the "DI" and is depicted in Figure 13 below. The XSD schema of the "DI" is provided in Table 8 of Annex A.2 below.

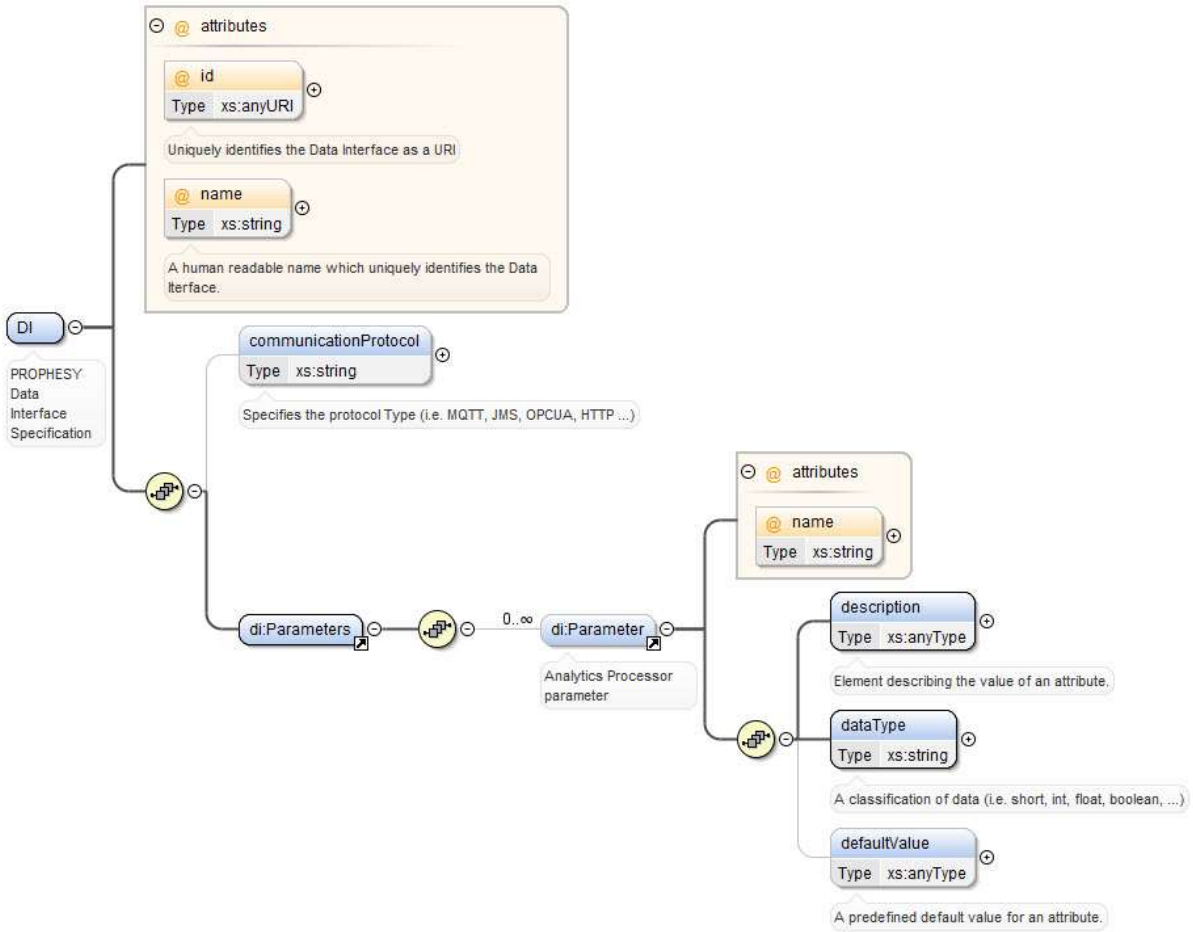


Figure 13: PROPHESY Data Interface entity

As depicted in Figure 13 above the "DI" has:

- **id**: an ID which uniquely identifies a Data Interface within a PROPHESY deployment
- **name**: A human-readable name which uniquely identifies the DataInterface

- **communicationProtocol**: Specifies the protocol Type (i.e. MQTT, JMS, OPCUA, HTTP ...).
- **parameters**: lists the required parameters to set up a specific communication interface. This entity includes:
  - **name**: A human-readable name which uniquely identifies the property for a specific DataInterface.
  - **description**: Provides an optional description of the property
  - **dataType**: A classification of data (i.e. short, int, float, boolean, ...)
  - **defaultValue**: An optional predefined default value for a property.

4.4.1.2.3 Data Source Definition (DSD)

This entity defines the properties of a data source in the shopfloor, such as a data stream from a sensor or an automation device. The root element of the Data Source Definition entity is the "DSD" and is depicted in Figure 14 below. The XSD schema of the "DSD" is provided in Table 6 of Annex A.2 below.

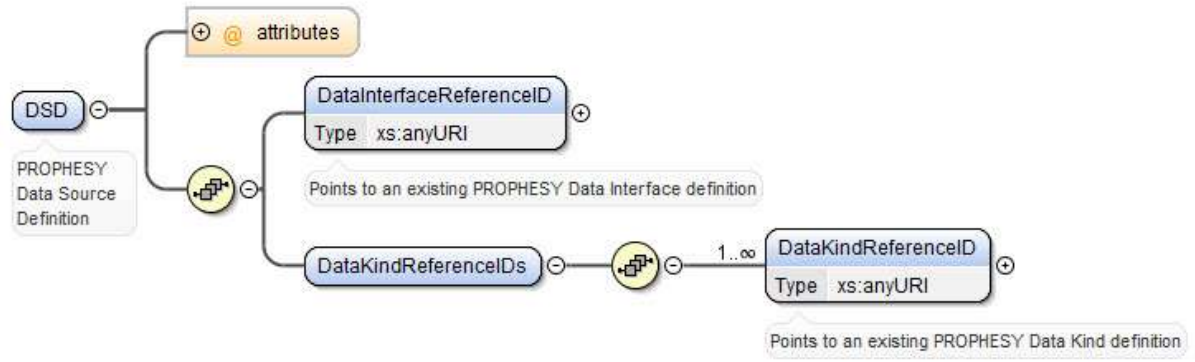


Figure 14: PROPHESY Data Source Definition entity

As depicted in Figure 14 above the "DSD" has:

- **id**: A required ID which uniquely identifies a Data Source Definition within a PROPHESY deployment.
- **name**: An optional human-readable name which uniquely identifies the Data Source Definition.
- **DataInterfaceReferenceID**: Points to an existing PROPHESY Data Interface definition
- **DataKindReferenceIDs**: contains a list of Data Kind IDs, which references to an existing PROPHESY Data Kind definition.

4.4.1.2.4 Processor Definition (PD)

This entity specifies a processing function to be applied on one or more data sources. It can be used to set up a data routing flow and to utilize analytics algorithms as well. The root element of the PROPHESY Processor Definition entity is the "PD" and is depicted in Figure 15 below. The XSD schema of the "PD" is provided in Table 11 of Annex A.2 below.

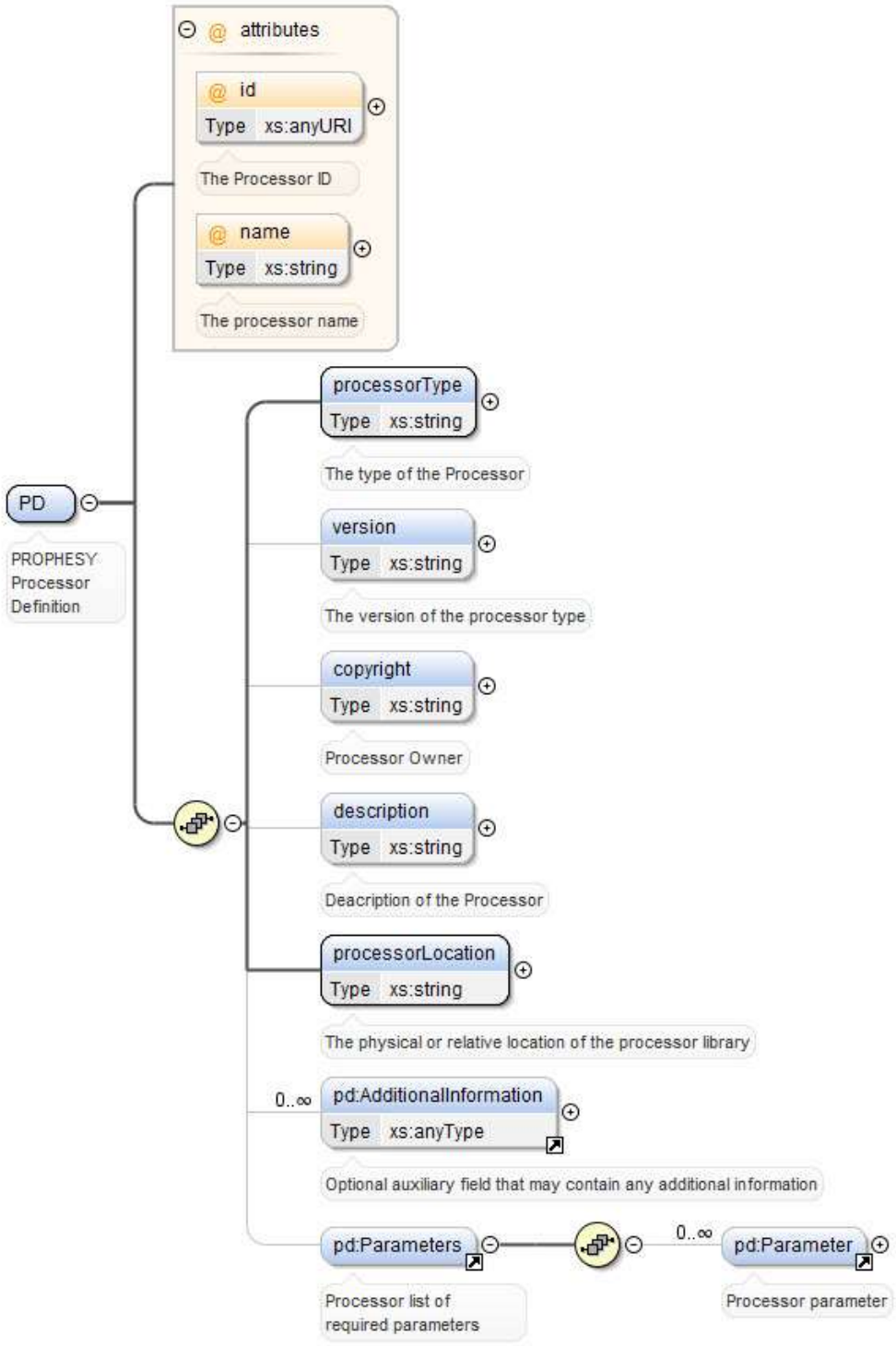


Figure 15: PROPHESY Processor Definition entity

As depicted in Figure 15 above the "PD" has:

- **id**: is the unique ID of a PROPHESY Processor Definition object.
- **name**: is an optional human recognisable name of the PROPHESY Processor Definition object.

- **processorType**: provides the processor type.
- **version**: provides the processor version information.
- **copyright**: which provides the processor ownership information.
- **description**: provide an optional description of the PROPHESY Processor Definition object.
- **processorLocation**: which provides the physical or relative location of the processor (i.e., a service endpoint or a fat application on a local path).
- **AdditionalInformation** (unlimited list): which provides an optional unlimited auxiliary field that may contain any additional information and it is used for further extensions. More specifically it serves as the root of the type definition hierarchy for any schema and it has the unique characteristic that it can function as a complex or a simple type definition, according to context. Additional information can be found in section 4.4.1.6.2 below
- **Parameters**: provides a list of Parameter entities which describes the required parameters that needs to be instantiated in order the processor to function. This entity is analysed with more details in the sections below.

Parameter

Processor Parameter entity describes a parameter which is required to be instantiated in order for a processor to run. i.e., a parameter could be a numerical value which can be used as input for a processor operation or a URL where a processor could push data streams. The root element of the Processor Parameters entity is the "Parameters" and is depicted in Figure 16 below. The XSD schema of the "Parameters" is provided in Table 11 of Annex A.2 below.

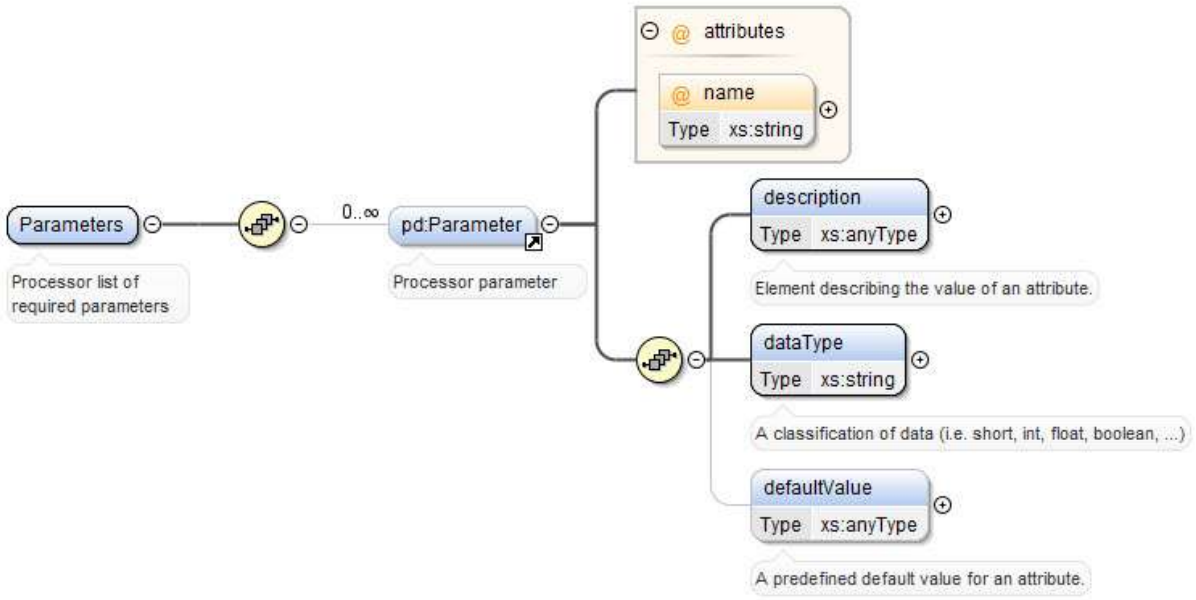


Figure 16: PROPHESY processor Parameters entity

As depicted in Figure 16 above the "Parameter" has:

- **name**: A human-readable name which uniquely identifies the property for a specific processor.
- **description**: Provides an optional description of the property.
- **dataType**: A classification of data (i.e. short, int, float, boolean, ...)
- **defaultValue**: An optional predefined default value for a property.

4.4.1.3 Data Manipulation

4.4.1.3.1 Processor Manifest (PM)

This entity represents an instance of a processors that is defined through the PD. The instance specifies the type of processors and its actual logic through linking to a programming function. The root element of the Processor Manifest entity is the "PM" and is depicted in Figure 17 below. The XSD schema of the "PM" is provided in Table 10 of Annex A.2 below.

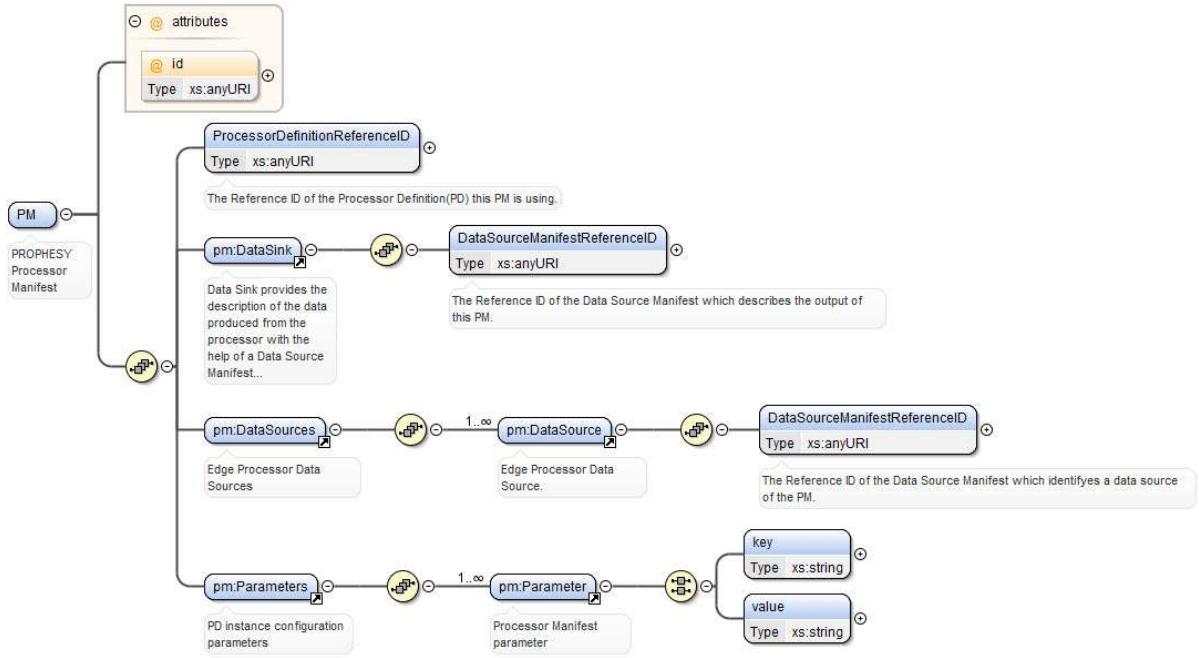


Figure 17: PROPHESY Processor Manifest entity

As depicted in Figure 17 above the "PM" has:

- **id**: a required unique identification of the Processor instance.
- **ProcessorDefinitionReferenceID**: The Reference ID of the Processor Definition(PD) this PM is used for instanciation.
- **DataSink**: Data Sink provides the description of the data produced from the processor with the help of a Data Source Manifest which it references. A PM is capable of producing one data source so in contains one:
  - **DataSourceManifestReferenceID**: The Reference ID of the Data Source Manifest which describes the output of this PM.

- **DataSources**: this entity provides the Processor instance data sources. A processor instance is capable of handling many data source so it includes a list of the following entities:
    - **DataSource**: specifies a single Processor Data Source this is achived by referencing an appropriate Data Source Manifest:
        - **DataSourceManifestReferenceID**: The Reference ID of the Data Source Manifest which identifies a data source of the PM.
- **Parameters**: this entity contains a list of the PD instance configuration parameters. The configuration parameters are consisted from a key-value pair where the key is specifies from the Processor Definition parameters this instance refers to (see ProcessorDefinitionReferenceID above)

4.4.1.3.2 Processor Orchestration (PO)

A PO entity represents an entire data routing workflow. It defines a combination of data processor instances (i.e. of PMs) that implements a distributed data routing task. The latter is likely to span multiple edge gateways (i.e., CPSs) and to operate over their data sources. The root element of the Processor Orchestrator entity is the "PO" and is depicted in Figure 18 below. The XSD schema of the "PO" is provided in Table 9 of Annex A.2 below.

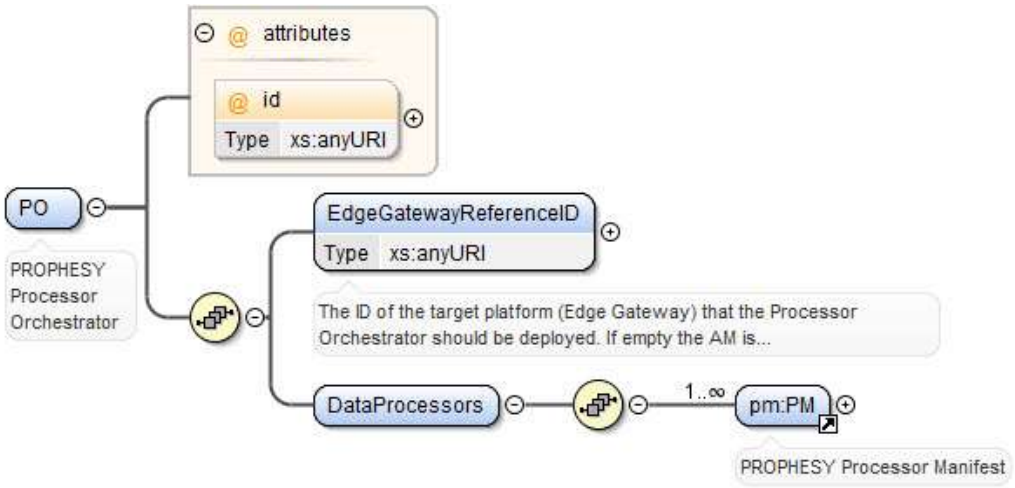


Figure 18: PROPHESY Processor Orchestrator entity

As depicted in Figure 18 above the "PO" has:

- **id**: a required unique identification of the Processor Orchestrator.
- **EdgeGatewayReferenceID**: The ID of the target platform (Edge Gateway) that the Processor Orchestrator should be deployed. If empty the AM is deployed locally (Directly to the Edge Gateway component).
- **DataProcessors**: contains a list of the Processor Manifests required to complete a data routing workflow. By grouping a list of PMs which specify their inputs and outputs and each PM is the consumer of other PMs this way we can define complex workflows. The workflow process is better explained in section 4.7 below.

4.4.1.4 Edge Gateway (CPS)

This entity models an edge gateway of a PROPHESY edge computing deployment focusing on the data aspects. In the scope of a PROPHESY deployment, data sources are associated with an edge gateway. This usually implies not only a logical association, but a physical association as well i.e. an edge gateway is deployed at a station and manages data sources in close physical proximity to the station. The root element of the Edge Gateway entity is the "EdgeGateway" and is depicted in Figure 19 below. The XSD schema of the "EdgeGateway" is provided in Table 4 of Annex A.2 below.

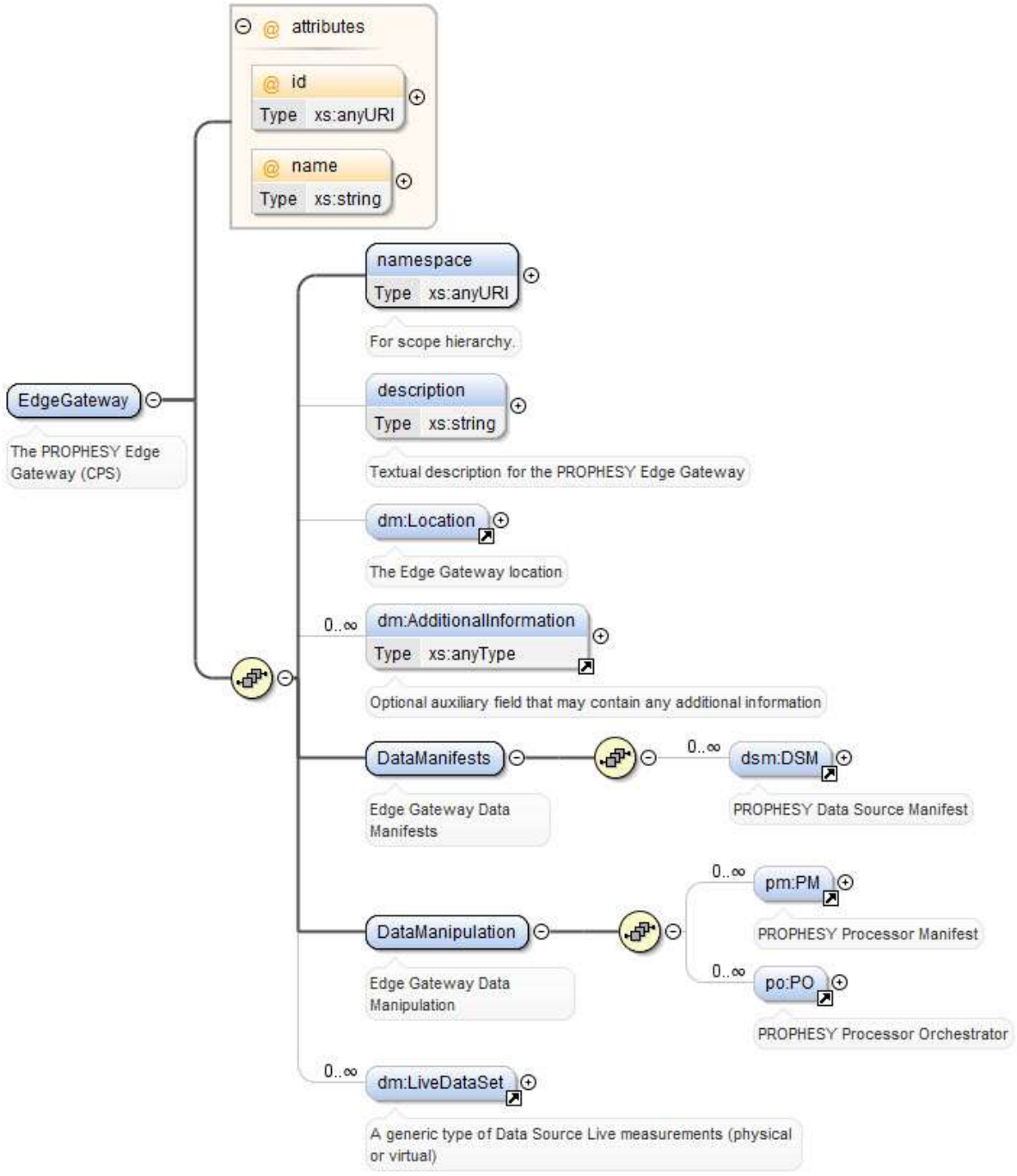


Figure 19: PROPHESY Edge Gateway (CPS) entity

As depicted in Figure 19 above the "EdgeGateway" has:

- **id**: A required ID which identifies the PROPHESY Digital Model instance.
- **name**: An optional human recognisable name of the PROPHESY Digital Model instance.
- **namespace**: the PROPHESY Edge Gateway which specifies the scope hierarchy.
- **description**: An optional textual description for the PROPHESY Edge Gateway.
- **Location**: The physical or virtual Edge Gateway location.
- **AdittionalInformation** (unlimited list): provides an optional unlimited auxiliary field that may contain any additional information and it is used for further extensions. More specifically, it serves as the root of the type definition hierarchy for any schema and it has the unique characteristic that it can function as a complex or a simple type definition, according to context. More details are provided in section 4.4.1.6.2 below.
- **DataManifests**: includes a list of instantiated DSD in the form of Data Source Manifests which describes data information along with the data interface. More information can be found in section 4.4.1.4.1 below.
- **DataManipulation**: Here the different reusable models capable of manipulating data are listed. These models are instantiation of models specified in the Data Definition section of PROPHESY DM root entity above and can be used locally (i.e., CPS). These Data manipulation entities should be synchronized with the PROPHESY DM root entity Data Definition section if it is required to be consumed globally (i.e., PdM) as well. More details are provided in section 4.4.1.3 above.
- **LiveDataSet**: the Live Data Set provides a structure capable of hosting the data streams and data sets of PROPHESY at the Edge Gateway level. More details are provided in section 4.4.1.5 below.

4.4.1.4.1 Data Source Manifest (DSM)

The DSM entity specifies a specific instance of a data source in-line with its DSD, DI and DK specifications. Multiple manifests (i.e. DSMs) are therefore used to represent the data sources that are available in the factory in the scope of the PROPHESY predictive maintenance. The root element of the Data Source Manifest entity is the "DSM" and is depicted in Figure 20 below. The XSD schema of the "DSM" is provided in Table 5 of Annex A.2 below.

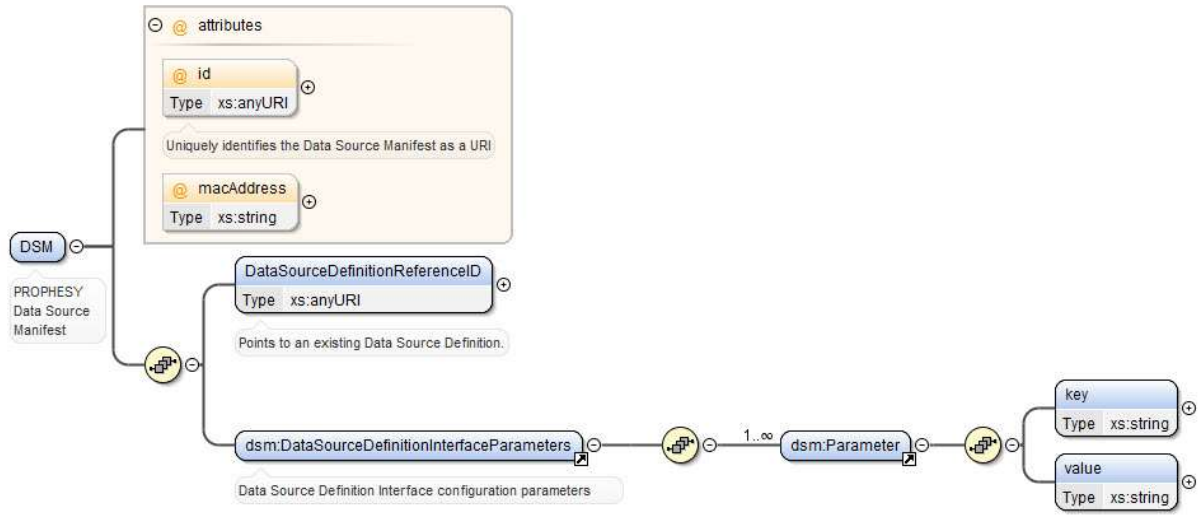


Figure 20: PROPHESY Data Source Manifest entity

As depicted in Figure 20 above the "DSM" has:

- id: A required ID which uniquely identifies the Data Source Manifest.
- macAddress: the optional mac address of the data source.
- DataSourceDefinitionReferenceID: Points to an existing Data Source Definition which is instantiated by adding the parameters below from this DSM.
- DataSourceDefinitionInterfaceParameters: contains the data Source Definition Interface configuration parameters in a key-value pair form. The Key is specified from the DI parameters referenced by the DSD which is referenced by the DSM.

4.4.1.5 Live Data Set (LDS)

The LDS entity models and represents the actual dataset that stem from an instance of a data source that is represented through a DSM. Hence, it references a DSM, which drives the specification of the types of the attributes of the LiveDataSet in-line with the DK. A LiveDataSet is associated with a timestamp and keeps track of the location of the data source in case it is associated with a mobile (rather than a stationary) data source. Hence, it has a location attribute as well. In principle the data source comprises a set of name-value pairs, which adhere to different data types in-line with the DK of the DSM. The root element of the Live Data Set entity is the "LiveDataSet" and is depicted in Figure 21 below. The XSD schema of the "LiveDataSet" is provided in Table 4 of Annex A.2 below.

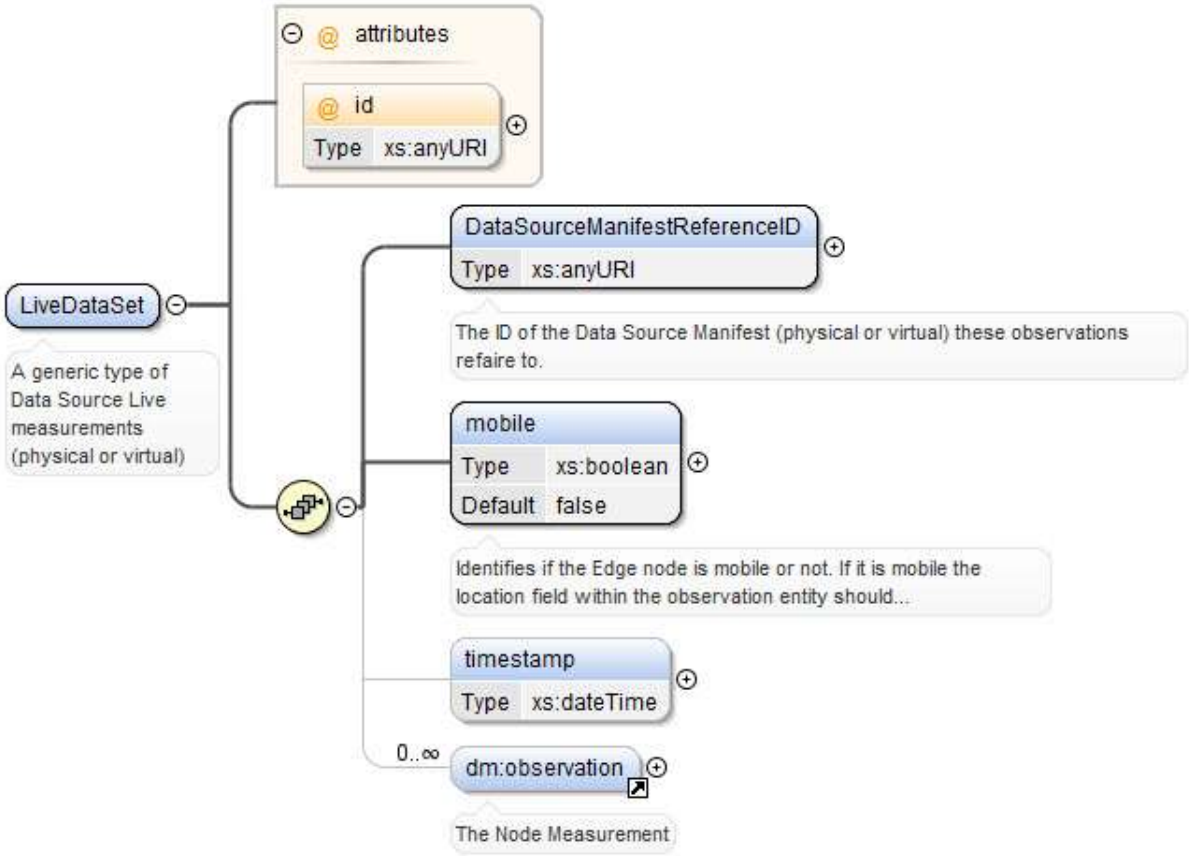


Figure 21: PROPHESY Live Data Set entity

As depicted in Figure 21 above Figure 23 the "LiveDataSet" has:

- **id**: A unique required ID which is assigned to every dataset when captured from the PROPHESY system.
- **DataSourceManifestReferenceID**: The ID of the Data Source Manifest (physical or virtual) these observations refer to.
- **mobile**: Identifies if the data source is mobile or not. If it is mobile the location field within the observation entity should be provided as well.
- **timestamp**: is the dateTime of when a batch of recorded observations was generated. If a data stream is observed (or only single values are observed) this field is not used and only the timestamp within the observation entity is used.
- **observations** (unlimited list): provides the value entity. This entity is analysed with more details in the section 4.4.1.5.1 below.

4.4.1.5.1 Observation

The observation entity provides a single measurement for a LiveDataSet. It provides information about the measurement type by referencing an existing Data Source Manifest, the timestamp of the observation and the location if the data source is specified as mobile.

The root element of a Live Data Set Observation entity is the “observation” and is depicted in Figure 22 below. The XSD schema of the “observation” is provided in Table 4 of Annex A.2 below.

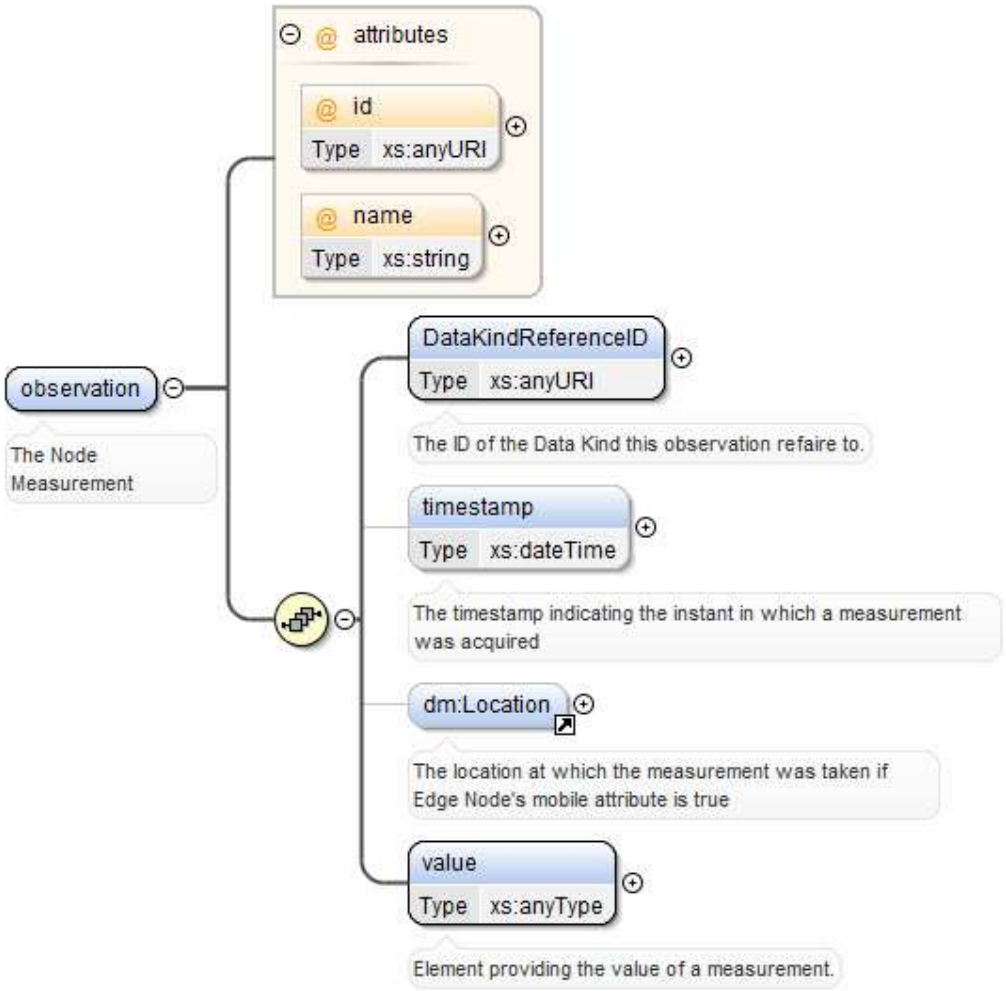


Figure 22: PROPHESY Live Data Set’s Observation entity

As depicted in Figure 22 above the “observation” has:

- **id**: which is the unique ID of the Observation instance.
- **name**: which is an optional humanly recognisable name of an observation.
- **DataKindReferenceID**: which provides information about the Observation supported Data Kind by referencing an existing DK instance.
- **timestamp**: which provides information about the dateTime an observation was recorded.
- **Location**: which provides the geographical or virtual location an incident took place if the data source is specified as mobile.
- **value**: which provides the value a Sensing and Acting connector observes.

4.4.1.6 Other Reusable Entities

4.4.1.6.1 Location

The root element of the Location model is the "Location" and is depicted in Figure 23 below. The XSD schema of the "Location" is provided in Table 4 of Annex A.2 below.

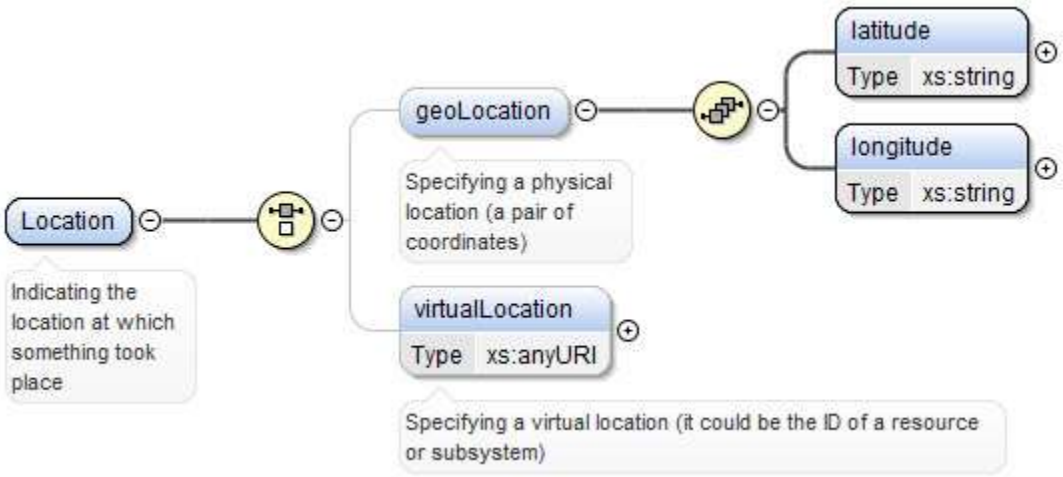


Figure 23: PROPHESY Location entity

As depicted in Figure 23 above the "Location" has:

- **geolocation**: which provides the coordinates (longitude and latitude) of a physical location.
- **virtualLocation**: which provides information about a virtual location (it could be the ID of a resource or subsystem).

4.4.1.6.2 Additional Information

AdditionalInformation is a generic entity which allows the extension of the existing data model with additional attributed that may be required. The root element of the "AdditionalInformation" model is depicted in Figure 24 below. The XSD schema of the "AdditionalInformation" is provided in Table 4 of Annex A.2 below.

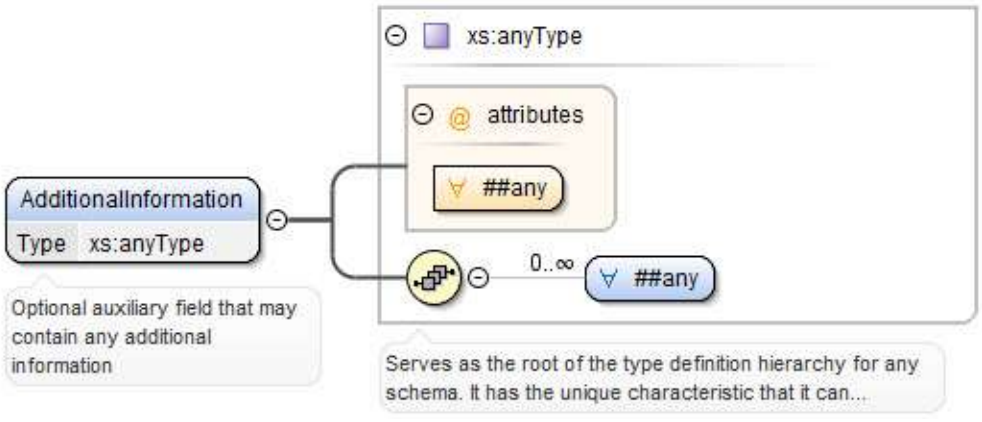


Figure 24: Additional Information entity

4.5 Maintenance Data Modelling

As already mentioned for the maintenance data a lightweight version of the MIMOSA CRIS (Common Relational Information Schema) from MANTIS project is going to be used. Figure 25 below illustrate the MANTIS class diagram of MIMOSA CRIS.

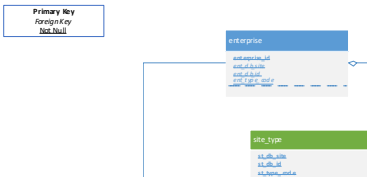


Figure 25: MANTIS class diagram of MIMOSA CRIS [10]

The supported classes depicted above are:

- as_remaining_life
- asset
- asset_blob_data
- asset_child
- meas_event
- meas_location
- mevent_blob_data
- mevent_chr_data
- mevent_num_data
- segment
- site
- sp_ampl_data
- sp_fft_data
- sp_time_data

More information can be found in MIMOSA CRIS documentation [9] and REST Interface for MIMOSA documentation from MANTIS [10].

4.6 Registry Data Modelling

As mentioned above PROPHESY will use the OpenO&M Common Interoperability Registry (CIR) as an Object registry model. CIR provides a standards-based, vendor-neutral method to map object entities belonging to different systems/databases which share common business context. Additionally, it:

- Enables the discoverability and relation of the registered objects and helps third party applications to combine the information provided from these systems/databases.
- Provides a global unique identifier (in a UUID format) for the registered objects.

The CIR provides an XML schema and a relational DB which describes the specification. In Figure 26 below, we can see the structure of the root elements of the Common Interoperability Registry. The OpenO&M (CIR) is open source and the latest version of the CIR schema can be found in MIMOSA organization GitHub[1].

As depicted in Figure 26 the "Registry" has [8]:

- **Registry**: The container object for a set of categories.
- **ID**: The user-defined identifier of the registry.
- **Description**: The description and expected use of the registry.
- **Category**: A Category object is the container object for a set of entries. Categories define sets of related or potentially related entries. For example, a Category may be defined for equipment hierarchy level names (Enterprise, Site, Area, Work Centre, Work Unit), which have alternate names on different systems. The combination of ID and SourceID must be unique within a Registry.

[1] https://github.com/mimosa-org/ws-cir/tree/master/xsd

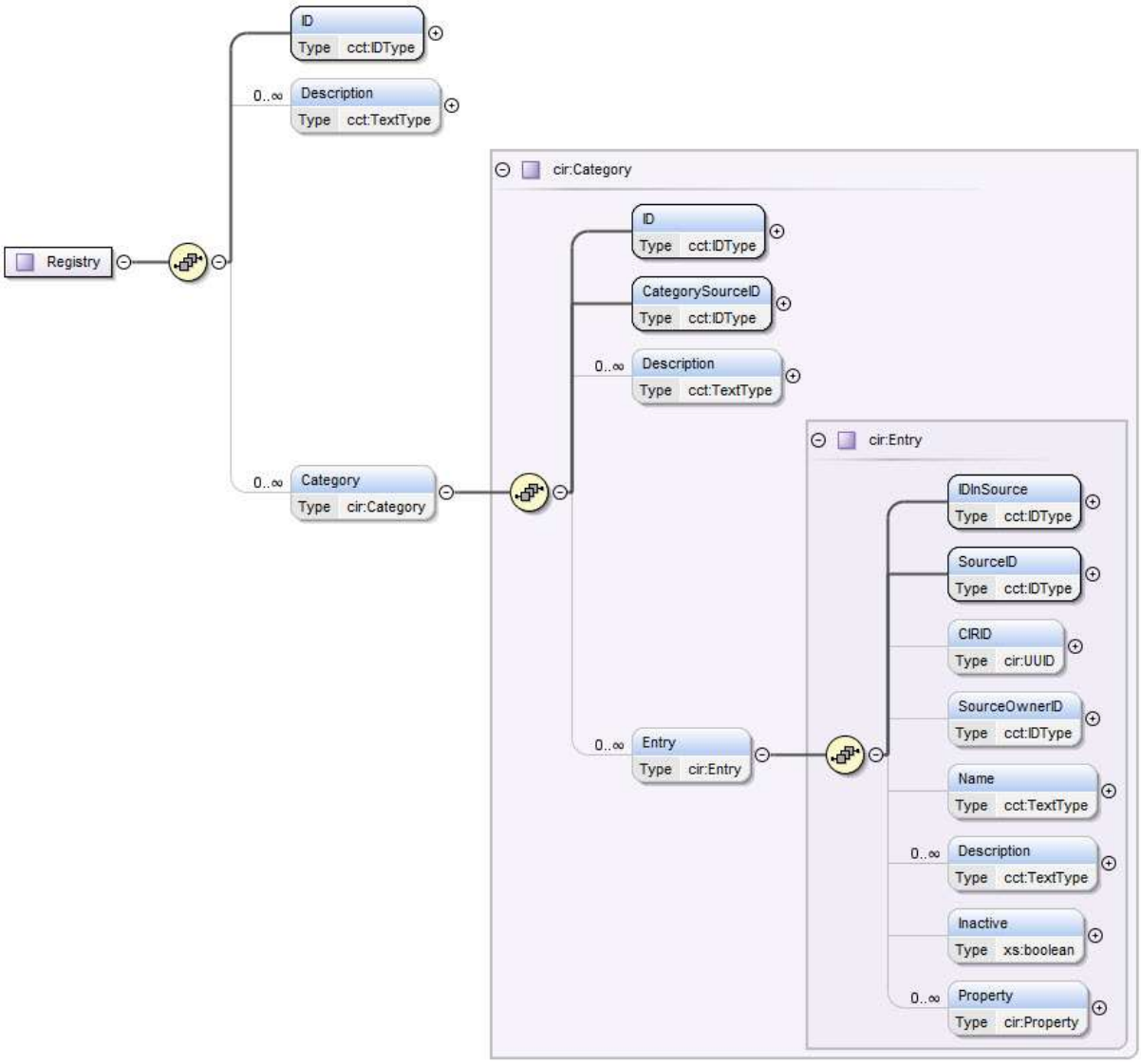


Figure 26: CIR Registry root entity

4.7 Data Routing

As mentioned in section 4.4.1.3.2 above the PROPHESY DM described above and more specifically the Data Manipulation part can be used for routing data from and to various data consumers and data produces. With the help of the PM and PO we can set up simple or complex data flows. A data flow example is depicted in Figure 27 below.

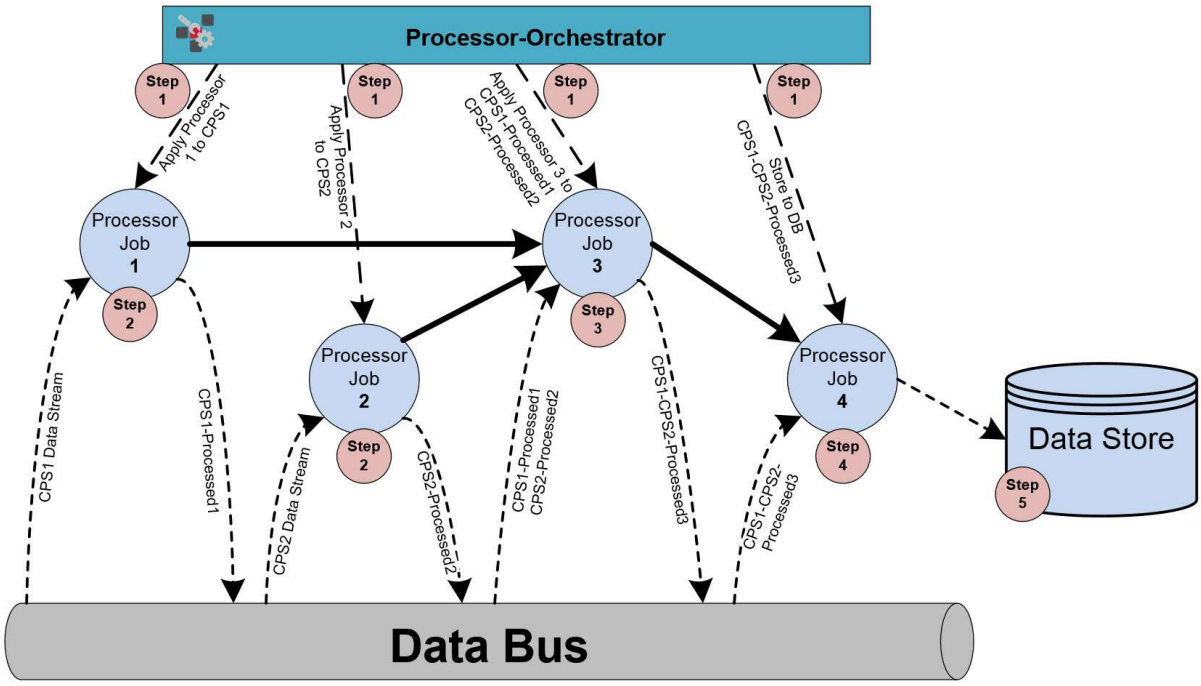


Figure 27: Data Routing Example

In Figure 27 above we can see two data streams (CPS1 and CPS2) which are pre-processed from Processor Job 1 (i.e. Pre-Processor) and Processor Job 2 (i.e. Pre-Processor) for an analytics algorithm (i.e. Processor Job 3) to be applied to them. The result is stored to a Data Storage with the help of Processor Job 4 (i.e. Storage Processor). The setup and runtime operation of the EA-Processor entails the following steps:

- Step1 (Set-up): Based on the description of the topology and required processors in the PM (Processor Manifest), the Processor Orchestrator instantiates and configures the required Processor jobs.
- Step2 (Runtime): Processor Job 1 consumes and pre-processes streams coming from CPS1. Likewise, Processor Job 2 consumes and pre-processes streams coming from CPS2.
- Step3 (Runtime): Processor Job 3 consumes the produced streams from Processor Job 1 and 2 for applying an analytics algorithm.
- Step4 (Runtime): Store Processor Job 4 consumes the data stream produced from Processor Job 3 and forwards it to the Data Storage.
- Step5 (Runtime): Data Storage persists the Data coming from Store Processor Job 4

5 Conclusions and Future Outlook

This deliverable has introduced the digital modelling for the PROPHESY predictive maintenance deployments. At the same time, it has introduced the PROPHESY-DM infrastructure as a means of modelling digital information for predictive maintenance, including relevant data/observations and metadata. PROPHESY-DM has been designed as a flexibly extensible system, which can be easily linked to other repositories of maintenance information. In particular, it provides the means for inter-linking maintenance datasets at the object level. Likewise, it allows maintenance information for a given asset to be fetched and accessed in an integrated fashion, regardless of the number and structure of the repositories that contain relevant information about the asset.

The PROPHESY-DM infrastructure has been designed in order to fulfil the requirements of the PROPHESY applications in terms of integrated and low-overhead access to maintenance datasets. At the same time, it has considered the available technologies of the project's partners, some of which have been reused in the scope of the first prototype implementation of the PROPHESY-DM infrastructure. In particular, the FAR-EDGE digital models have been adapted to PROPHESY needs and provide the means for modelling and managing streaming data sources as part of the PROPHESY-DM platform. Likewise, the lightweight MIMOSA database that was developed in the MANTIS project provides the means for specifying and accessing maintenance related metadata in the scope of the PROPHESY applications.

The PROPHESY-DM implementation that accompanies this deliverable is the first step to producing the complete digital modelling vision of the project. Some features (e.g., the management console) are not yet implemented and will be provided in the second version of the deliverable (i.e. deliverable D3.4). Likewise, no repositories of maintenance data have been lined so far in order to validate the proper operation of the interoperability registry. These features will be implemented as part of the project's development roadmap for the second year of the project. In the coming months this roadmap will be also enhanced with features stemming from feedback about integrating and use of the CIR in the PROPHESY platform. Therefore, the final version of the PROPHESY-DM infrastructure (that will be part of deliverable D3.4, will contain both enhancement of the infrastructure with new functionalities and fine-tuning of the existing functionalities based on stakeholders' feedback from WP4, WP5 and WP7 of the project.

6 References

[1] W. L. B, A. Lobato-jimenez, and E. Axinia, “A Survey on Standards and Ontologies for Process Automation,” vol. 9266, pp. 22–32, 2015.

[2] R. S. Peres, M. Parreira-Rocha, A. D. Rocha, J. Barbosa, P. Leitão and J. Barata, "Selection of a data exchange format for industry 4.0 manufacturing systems," IECON 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, 2016, pp. 5723-5728, doi: 10.1109/IECON.2016.7793750.

[3] Frederik Gosewehr, Jeffrey Wermann, Waldemar Borsych, Armando Walter Colombo, "Specification and design of an industrial manufacturing middleware", Industrial Informatics (INDIN) 2017 IEEE 15th International Conference on, pp. 1160-1166, 2017.

[4] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, “AutomationML: From data exchange to system planning and simulation,” IEEE International Conference on Industrial Technology (ICIT), 2012, pp. 378–383.

[5] MIMOSA OSA-CBM, “Open System Architecture for Condition-Based Maintenance”, available at: http://www.mimosa.org/mimosa-osa-cbm

[6] Mathew, Avin D. and Zhang, Liqun and Zhang, Sheng and Ma, Lin (2006) A review of the MIMOSA OSA-EAI database for condition monitoring systems. In Proceedings World Congress on Engineering Asset Management, Gold Coast, Australia.

[7] A Muller, M C Suhner, B lung, 'Maintenance alternative integration to prognosis process engineering', Journal of Quality in Maintenance Engineering, Vol 13, Iss 2, pp 198-211, 2007.

[8] Avin Mathew, Ken Bever, Dennis Brandl, “Web Service Common Interoperability Registry 1.0”, OpenO&M Candidate Standard 19 June 2015, available at: http://www.openoandm.org/ws-cir/1.0/ws-cir.html

[9] MIMOSA Org, “Common Relational Information Schema (CRIS) Specification”, Version 3.2.3, 2012

[10] Juha Valtonen (LUAS), Antti Niemelä (LUAS), Iñaki Arenaza (MGEP), “REST Interface for MIMOSA Documentation”, 13/06/2017

Appendix A Data Models

A.1 Schema Specification

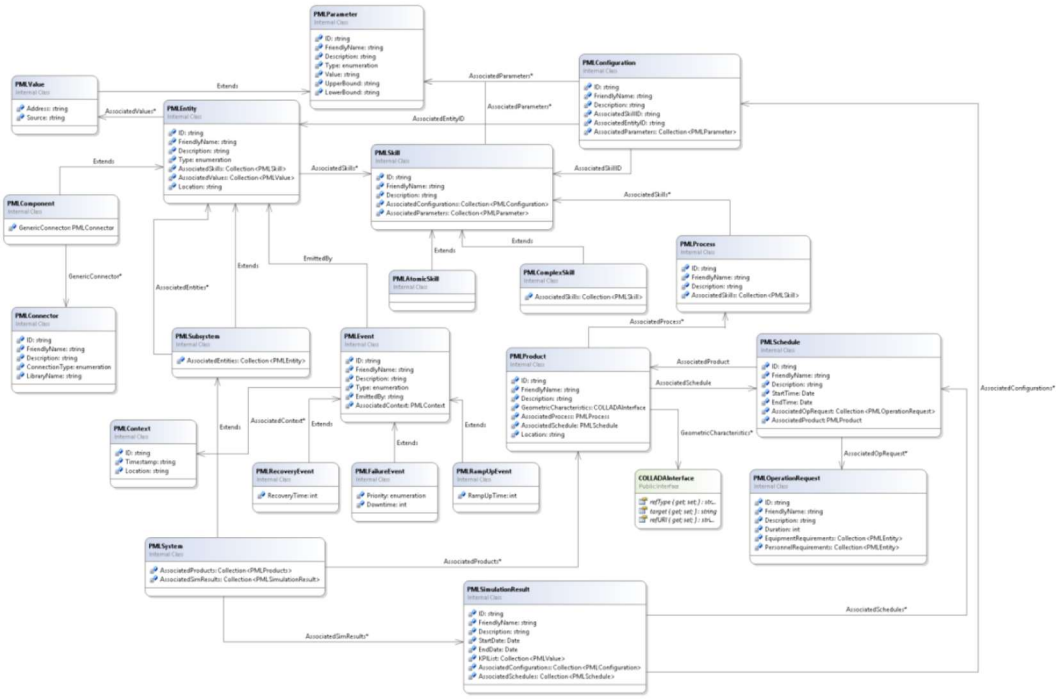


Figure 28: PERFoRMML Class Diagram

A.2 PROPHESY DM Schemata

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="eu:prophesy:prophesy:dm" elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:dm="eu:prophesy:prophesy:dm"
  xmlns:dcd="eu:prophesy:drpp:dcd" xmlns:dsm="eu:prophesy:drpp:dsm"
  xmlns:dcm="eu:prophesy:drpp:dcm" xmlns:pd="eu:prophesy:pd"
  xmlns:dsd="eu:prophesy:drpp:dsd" xmlns:dk="eu:prophesy:drpp:dk"
  xmlns:di="eu:prophesy:drpp:di" xmlns:po="eu:prophesy:processor:orchestrator"
  xmlns:pm="eu:prophesy:pm">


  <xs:import namespace="eu:prophesy:drpp:dk" schemaLocation="DataKind.xsd" />
  <xs:import namespace="eu:prophesy:drpp:di" schemaLocation="DataInterface.xsd" />
  <xs:import namespace="eu:prophesy:drpp:dsd"
                  schemaLocation="DataSourceDefinition.xsd" />
  <xs:import namespace="eu:prophesy:drpp:dsm" schemaLocation="DataSourceManifest.xsd" />
```

```
<xs:import namespace="eu:prophesy:pd" schemaLocation="ProcessorDefinition.xsd" />
<xs:import namespace="eu:prophesy:processor:orchestrator"
  schemaLocation="ProcessorOrchestrator.xsd" />
<xs:import namespace="eu:prophesy:pm" schemaLocation="ProcessorManifest.xsd" />

<xs:element name="ProphesyDM">
  <xs:annotation>
    <xs:documentation>PROPHESY Digital Model</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="namespace" type="xs:anyURI">
        <xs:annotation>
          <xs:documentation>For scope hierarchy.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="description" minOccurs="0" type="xs:string">
        <xs:annotation>
          <xs:documentation>Textual description for the PROPHESY Digital
            Model
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="DataDefinitions">
        <xs:annotation>
          <xs:documentation>Digital Models Data Definitions</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="dsd:DSD" />
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="di:DI" />
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="dk:DK" />
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="pd:PD" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="DataManipulation">
        <xs:annotation>
          <xs:documentation>Digital Models Data Manipulation
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="pm:PM" />
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="po:PO" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element maxOccurs="unbounded" minOccurs="0"
        ref="dm:AdditionalInformation" />
      <xs:element ref="dm:EdgeGateway" maxOccurs="unbounded"
        minOccurs="0" />
      <xs:element ref="dm:LiveDataSet" maxOccurs="unbounded"
        minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:anyURI" />
    <xs:attribute name="name" type="xs:string" />
  </xs:complexType>
</xs:element>
```

```
<xs:element name="EdgeGateway">
  <xs:annotation>
    <xs:documentation>The PROPHESY Edge Gateway (CPS)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="namespace" type="xs:anyURI">
        <xs:annotation>
          <xs:documentation>For scope hierarchy.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="description" minOccurs="0" type="xs:string">
        <xs:annotation>
          <xs:documentation>Textual description for the PROPHESY Edge
            Gateway
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element minOccurs="0" ref="dm:Location">
        <xs:annotation>
          <xs:documentation>The Edge Gateway location</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element maxOccurs="unbounded" minOccurs="0"
        ref="dm:AdditionalInformation" />
      <xs:element name="DataManifests">
        <xs:annotation>
          <xs:documentation>Edge Gateway Data Manifests</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="dsm:DSM"
              minOccurs="0" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="DataManipulation">
        <xs:annotation>
          <xs:documentation>Edge Gateway Data Manipulation</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="pm:PM" />
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="po:PO" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element ref="dm:LiveDataSet" maxOccurs="unbounded"
        minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:anyURI" />
    <xs:attribute name="name" type="xs:string" />
  </xs:complexType>
</xs:element>

<xs:element name="LiveDataSet">
  <xs:annotation>
    <xs:documentation>A generic type of Data Source Live measurements
      (physical or virtual)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="DataSourceManifestReferenceID" type="xs:anyURI">
```

```
          <xs:annotation>
            <xs:documentation>The ID of the Data Source Manifest (physical or
              virtual) these
              observations refaire to.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element default="false" name="mobile" type="xs:boolean">
          <xs:annotation>
            <xs:documentation>Identifies if the Edge node is mobile or not. If
              it is mobile the location field within the observation entity
              should be provided as well.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element minOccurs="0" name="timestamp" type="xs:dateTime" />
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="dm:observation" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>

  <xs:element name="observation">
    <xs:annotation>
      <xs:documentation>The Node Measurement</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DataKindReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>The ID of the Data Kind this observation refaire
              to.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element form="qualified" name="timestamp" type="xs:dateTime"
            minOccurs="0">
          <xs:annotation>
            <xs:documentation>The timestamp indicating the instant in which
              a measurement was acquired
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="dm:Location" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The location at which the measurement was
              taken if Edge Node's mobile attribute is true
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="value" type="xs:anyType" minOccurs="1">
          <xs:annotation>
            <xs:documentation>Element providing the value of a measurement.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI" />
      <xs:attribute name="name" type="xs:string" />
    </xs:complexType>
  </xs:element>

  <xs:element name="Location">
    <xs:annotation>
      <xs:documentation>Indicating the location at which something took place
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice>
        <xs:element maxOccurs="1" minOccurs="0" name="geoLocation">
          <xs:annotation>
            <xs:documentation>Specifying a physical location (a pair of
              coordinates) </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="latitude" type="xs:string" />
              <xs:element name="longitude" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element minOccurs="0" name="virtualLocation" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>Specifying a virtual location (it could be the ID
              of a resource or subsystem)</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:element name="AdditionalInformation" type="xs:anyType">
    <xs:annotation>
      <xs:documentation>Optional auxiliary field that may contain any additional
        information
      </xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

Table 4: PROPHESY DM schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:drpp:dsm"
  xmlns:dsm="eu:prophesy:drpp:dsm" xmlns:dsd="eu:prophesy:drpp:dsd">

  <xs:import namespace="eu:prophesy:drpp:dsd"
                                    schemaLocation="DataSourceDefinition.xsd" />



  <xs:element name="DSM">
    <xs:annotation>
      <xs:documentation>PROPHESY Data Source Manifest</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DataSourceDefinitionReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>Points to an existing Data Source Definition.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="dsm:DataSourceDefinitionInterfaceParameters" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI">
```

```
        <xs:annotation>
          <xs:documentation>Uniquely identifies the Data Source Manifest as a
            URI</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="macAddress" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element name="DataSourceDefinitionInterfaceParameters">
    <xs:annotation>
      <xs:documentation>Data Source Definition Interface configuration
        parameters</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="dsm:Parameter" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Parameter">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="key" />
        <xs:element type="xs:string" name="value" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 5: Data Source Manifest (DSM) schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:drpp:dsd"
  xmlns:dsd="eu:prophesy:drpp:dsd" xmlns:dk="eu:prophesy:drpp:dk"
  xmlns:di="eu:prophesy:drpp:di">

  <xs:element name="DSD">
    <xs:annotation>
      <xs:documentation>PROPHESY Data Source Definition</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DataInterfaceReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>Points to an existing PROPHESY Data Interface
              definition</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="DataKindReferenceIDs">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DataKindReferenceID" type="xs:anyURI"
                maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>Points to an existing PROPHESY Data Kind
                    definition</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

```
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI">
        <xs:annotation>
          <xs:documentation>Uniquely identifies the Data Source as a URI
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="xs:string">
        <xs:annotation>
          <xs:documentation>A human readable name which uniquely identifies the
            Data Source.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 6: Data Source Definition (DSD) schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:drpp:dk"
  xmlns:dk="eu:prophesy:drpp:dk">

  <xs:element name="DK">
    <xs:annotation>
      <xs:documentation>PROPHESY Data Kind</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="description" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Provides an Optional description of the Data Kind
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element type="xs:string" name="modelType" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Specifies the Model type of the Data (i.e. SenML,
              OM, ...)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="format" type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Specifies the format of the Data (i.e. JSON,
              XML,...)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="quantityKind" minOccurs="0" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>A QuantityKind is an abstract classifier that
              represents the concept of "kind of quantity". A QuantityKind
              represents the essence of a quantity without any numerical value
              or unit. (e.g. A sensor -sensor1- measures temperature: sensor1
              has quantityKind temperature) </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI">
        <xs:annotation>
          <xs:documentation>Uniquely identifies the Data Kind as a URI
          </xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="xs:string">
        <xs:annotation>
          <xs:documentation>A human readable name which uniquely identifies the
            Data Kind.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Table 7: Data Kind (DK) schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:drpp:di"
  xmlns:di="eu:prophesy:drpp:di">

  <xs:element name="DI">
    <xs:annotation>
      <xs:documentation>PROPHESY Data Interface Specification</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="communicationProtocol"
          minOccurs="0">
          <xs:annotation>
            <xs:documentation>Specifies the protocol Type (i.e. MQTT, JMS,
              OPCUA, HTTP ...)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="di:Parameters" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI">
        <xs:annotation>
          <xs:documentation>Uniquely identifies the Data Interface as a URI
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="xs:string">
        <xs:annotation>
          <xs:documentation>A human readable name which uniquely identifies the
            Data Iterface.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="Parameters">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="di:Parameter" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Parameter">
    <xs:annotation>
      <xs:documentation>Analytics Processor parameter</xs:documentation>
    </xs:annotation>
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="description" type="xs:anyType" minOccurs="1">
          <xs:annotation>
            <xs:documentation>Element describing the value of an attribute.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element minOccurs="1" name="dataType" type="xs:string">
          <xs:annotation>
            <xs:documentation>A classification of data (i.e. short, int, float,
              boolean, ...)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="defaultValue" type="xs:anyType" minOccurs="0">
          <xs:annotation>
            <xs:documentation>A predefined default value for an attribute.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 8: Data Interface (DI) schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="eu:prophesy:processor:orchestrator" elementFormDefault="qualified"
  xmlns:am="eu:prophesy:processor:orchestrator" xmlns:pm="eu:prophesy:pm">

  <xs:import namespace="eu:prophesy:pm" schemaLocation="ProcessorManifest.xsd" />

  <xs:element name="PO">
    <xs:annotation>
      <xs:documentation>PROPHESY Processor Orchestrator</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EdgeGatewayReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>The ID of the target platform (Edge Gateway) that
              the Processor Orchestrator should be deployed. If empty the AM is
              deployed locally (Directly to the Edge Gateway component).
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="DataProcessors">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" ref="pm:PM" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 9: Processor Orchestrator (PO) schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:pm"
  xmlns:pm="eu:prophesy:pm"  xmlns:pd="eu:prophesy:pd">

  <xs:import namespace="eu:prophesy:pd" schemaLocation="ProcessorDefinition.xsd" />

  <xs:element name="PM">
    <xs:annotation>
      <xs:documentation>PROPHESY Processor Manifest</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProcessorDefinitionReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>The Reference ID of the Processor Definition(PD)
              this PM is using.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="pm:DataSink" />
        <xs:element ref="pm:DataSources" />
        <xs:element ref="pm:Parameters">
          <xs:annotation>
            <xs:documentation>PD instance configuration parameters
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
  <xs:element name="DataSink">
    <xs:annotation>
      <xs:documentation>Data Sink provides the description of the data produced
        from the processor with the help of a Data Source Manifest which it
        references.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DataSourceManifestReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>The Reference ID of the Data Source Manifest which
              describes the output of this PM.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DataSources">
    <xs:annotation>
      <xs:documentation>Edge Processor Data Sources</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="pm:DataSource" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DataSource">
```

```
    <xs:annotation>
      <xs:documentation>Edge Processor Data Source.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DataSourceManifestReferenceID" type="xs:anyURI">
          <xs:annotation>
            <xs:documentation>The Reference ID of the Data Source Manifest which
              identifyes a data source of the PM.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Parameters">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="pm:Parameter" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Parameter">
    <xs:annotation>
      <xs:documentation> Processor Manifest parameter</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:all>
        <xs:element name="key" type="xs:string" />
        <xs:element name="value" type="xs:string" />
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 10: Processor Manifest (PM) schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="eu:prophesy:pd"
  xmlns:pd="eu:prophesy:pd">

  <xs:element name="PD">
    <xs:annotation>
      <xs:documentation>PROPHESY Processor Definition</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="processorType"
          type="xs:string">
          <xs:annotation>
            <xs:documentation>The type of the Processor</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="version" type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The version of the processor type
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="copyright" type="xs:string" maxOccurs="1"
          minOccurs="0">
```

```
          <xs:annotation>
            <xs:documentation>Processor Owner</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element minOccurs="0" maxOccurs="1" name="description"
          type="xs:string">
          <xs:annotation>
            <xs:documentation>Deacription of the Processor</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element minOccurs="1" maxOccurs="1" name="processorLocation"
          type="xs:string">
          <xs:annotation>
            <xs:documentation>The physical or relative location of the processor
              library</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element maxOccurs="unbounded" minOccurs="0"
          ref="pd:AdditionalInformation" />
        <xs:element minOccurs="0" maxOccurs="1" ref="pd:Parameters" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:anyURI" use="optional">
        <xs:annotation>
          <xs:documentation>The Processor ID</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="xs:string">
        <xs:annotation>
          <xs:documentation>The processor name</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="Parameters">
    <xs:annotation>
      <xs:documentation> Processor list of required parameters
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="pd:Parameter" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Parameter">
    <xs:annotation>
      <xs:documentation> Processor parameter</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="description" type="xs:anyType" minOccurs="1">
          <xs:annotation>
            <xs:documentation>Element describing the value of an attribute.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element minOccurs="1" name="dataType" type="xs:string">
          <xs:annotation>
            <xs:documentation>A classification of data (i.e. short, int, float,
              boolean, ...)</xs:documentation>
          </xs:annotation>
```

```
        </xs:element>
        <xs:element name="defaultValue" type="xs:anyType" minOccurs="0">
          <xs:annotation>
            <xs:documentation>A predefined default value for an attribute.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element name="AdditionalInformation" type="xs:anyType">
    <xs:annotation>
      <xs:documentation>Optional auxiliary field that may contain any additional
        information</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

Table 11: Processor Definition (PD) schema