DENiM

Deliverable D3.4
DENiM Architecture Specification and Semantic Models

Date: 22.07.2021
Version: R1.0
Dissemination Level: PU

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Project Number	:	958339
Project Title	:	Digital intelligence for collaborative Energy Management in Manufacturing – DENiM
Deliverable Dissemination Level	:	PU

Deliverable Number	:	D3.4
Title of Deliverable	:	DENiM Architecture Specification and Semantic Models
Nature of Deliverable	:	Report
Internal Document Number	:	D3.4
Contractual Delivery Date	:	31st July 2021
Actual Delivery Date	:	22nd July 2021
Work Package	:	WP3
Author(s)	:	F. Dorado (IDE), J. Kortelainen (VTT), C. Leyva(IDE), F. Pacull (BAG), L. Parrilla (IDE), A. McGibney (MTU)
Total number of pages (including cover)	:	63

Executive Summary

Deliverable 3.4 describes a review of semantic data modelling to support interoperability and data sharing as well as the specification of the core data architecture for the DENiM project. Both sections are organized as follows: requirements and analysis of existing approaches and presentation of the selected approach and technical specification.

The design of both the semantics and the architecture have been done with the intention of making it usable across a diverse set of industries. While guided by the DENiM project requirements, it should be acknowledge that the approach taken is to create a more general and exploitable architecture that enables interoperability and data sharing across a wide range of sectors. An extensive analysis of the project needs was done in collaboration with the tool developers to describe their designs in terms of data interaction. With the outputs of these documents, a list of relevant fields considered for semantic models and a set of requirements for the architecture design were generated.

The outputs of this deliverable, together with requirements captured as part of D3.1, focused on the pilots' needs, will form a basis for the evolution from requirements and specification to components design and development.

This document may be reviewed or updated if and where necessary throughout the development lifecycle.

Keyword list

Data Architecture, Services, Requirements, Specification, Semantics, Ontologies

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Document History

Date	Revision	Comment	Author/Editor	Affiliation
01.02.2021	V0.1	Draft	Lidia Parrilla	IDE
05.03.2021	V0.1	First draft of sections 2.1 and 2.2.1.	Juha Kortelainen	VTT
31.03.2021	V0.1	Completed the first version of the section for semantic tools and platforms. Reorganised section 2.2.1.	Juha Kortelainen	VTT
05.05.2021	V0.1	Reorganisation of chapter 3 to start documenting the IT architecture specification	Lidia Parrilla	IDE
10.05.2021	V0.2	Transferred contents to the deliverable template.	Lidia Parrilla	IDE
01.06.2021	V0.2	V0 of section 3.8.4	François Pacull	BAG
14.06.2021	V0.2	Reviewed 3.8.4, completed several parts of semantics, full review, added acronyms.	Lidia Parrilla	IDE
17.06.2021	V0.3	Reviewed whole deliverable	Carlos Leyva	IDE
21.06.2021	V0.3	Added section about Architecture Manager service	Lidia Parrilla	IDE
25.06.2021	V0.4	Conclusions and executive summary	Lidia Parrilla	IDE
29.06.2021	V0.5	Review of chapter 3	Fernando Dorado	IDE
30.06.2021	V0.6	Full review	Lidia Parrilla	IDE
05.07.2021	V0.6	Internal Review	Susan Rea	MTU
08.07.2021	V0.7	First-round review comments addressed	Lidia Parrilla	IDE
14.07.2021	V0.8	Full review	Fabio Rodríguez	USE
15.07.2021	V1.0	All review comments addressed and versions merged. Ready for submission	Lidia Parrilla	IDE
22.07.2021	R1.0	Final Review and release for submission	A. McGibney	MTU

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Table of Contents

1 Introduction 8
2 Semantic modelling 9
2.1 Introduction to semantic modelling 9
2.2 Semantic requirements in DENiM 11
2.3 Existing approaches, concepts and technologies 13
2.3.1 Semantic Web technologies 13
2.3.2 Energy-focused semantic models and ontologies for manufacturing 19
2.3.3 Semantics for data modelling and modelling of systems 25
2.4 DENIM's semantic approach 27
2.4.1 SAREF ontology with the DENiM platform 28
3 DENiM Digital Intelligence Platform: Architecture specification 29
3.1 DENiM architecture requirements 29
3.2 DENiM architecture design 30
3.2.1 Manufacturing site 30
3.2.2 Data acquisition & Interoperability 32
3.2.3 Digital Intelligence Platform 32
3.3 DENiM data management architecture 34
3.3.1 Current practices on data platforms 34
3.3.2 Data architecture requirements 35
3.3.3 Data architecture specification 37
3.3.4 Platform services 49
3.4 Industry 4.0 reference architectures and mapping to the DENiM platform 57
4 Conclusion 59

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Glossary

AAS	Asset Administration Shell
ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ARM	Assembly Relation Model
AsD	Assembly Design
C2IEDM	Command and Control Information Exchange Data Model
CoAP	Constrained Application Protocol
COSMO	Component Simulation and Modeling Ontology
CQL	Cassandra Query Language
DeMO	Discrete-event Modeling Ontology
DENiM	Digital intelligence for collaborative ENergy management in Manufacturing
DEVS	Discrete-event Systems Specification
DoS	Denial of Service
DT	Digital Twin
EDL	Eclipse Distribution License
EnPI	Energy Performance Indicator
ERP	Entrerprise Resource Planning
ETSI	European Telecommunications Standards Institute
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
HDFS	Hadoop Distributed File System
HTTP	Hypertext Transfer Protocol
HVAC	Heating, ventilation, and air conditioning
I/O	Input/Output
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IIRA	Industrial Internet Reference Architecture

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KE	Knowledge Engineering
LA	Lambda Architecture
LCA	Lifecycle Analysis
LCCA	Lifecycle Cost Analysis
MASON	MAnufacturing's Semantics ONtology
MES	Manufacturing Execution System
ML	Machine Learning
MLOps	Machine Learning Operations
MOF	Meta Object Facility
MQTT	Message Queue Telemetry Transport
MUO	Measurements Units Ontology
OAM	Open Assembly Model
OLAP	Online analytical processing
OMG	Object Management Group
ONNX	Open Neural Network Exchange
OPC UA	Open Communication Standard Unified Architecture
OS	Operating System
OWA	Open World Assumption
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logic
PIMODES	Process Interaction Modeling Ontology for Discrete-event Simulations
PLC	Programmable Logic Controller
RAMI 4.0	Reference Architecture Model for Industry 4.0
RDF	Resource Description Framework
RDF-S	Resource Description Framework - Schema
RIF	Rule Interchange Format
SAREF	Smart Applications REFerence
SCADA	Supervisory control and data acquisition
SDM	Semantic Data Model
SHACL	Shapes Constraint Language

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

SQL	Structured Query Language
SSD	Solid State Disk
SSN	Semantic Sensors Network
STEP	Space Time Entity and Process
SWRL	Semantic Web Rule Language
SysML	Systems Modeling Language
TD	Thing Description
UI	User Interface
UML	Unified Modelling Language
UNA	Unique Name Assumption
W3C	World Wide Web Consortium
WoT	Web of Things
XML	Extensive Markup Language
XSLT	Extensible Stylesheet Language Transformations
YAML	Yet Another Markup Language
YARN	Yet Another Resource Negotiator

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

1 Introduction

This deliverable corresponds to the outputs of tasks 3.4 and 3.5 of the DENiM project, which refer to the semantic modelling and the data architecture specification, respectively. These two concepts are interlinked with semantic modelling being a critical matter to be taken into account for the architecture design. This is to ensure that the DENiM platform is extensible and usable across a wide range of industries and sectors. Data from many different sources need to be extracted, combined and used as input for diverse applications, tools and services. This means that the data architecture will collect data that stems from separate fields, in different formats, units, etc. and at the same time will produce data to be consumed by services, tools or directly by users in a variety of formats. For this reason, the focus is on integrating semantic services and tools within the platform to ensure a common understanding of the meaning of managed data, as well as supporting interoperability, reusability and information exchange among developers, operators, and diverse end-users.

For these reasons, both the architecture specification and semantic models are considered together as part of this deliverable. Chapter 2 is dedicated to semantics and contains an initial introduction to the basic concepts in semantics, an analysis of the semantic requirements for the DENiM project, followed by a study of the existing approaches, ontologies, and technologies. After this, the selected semantic approach for the DENiM project is presented.

Next, Chapter 3 presents the DENiM data architecture and is divided into four subsections. First, the high-level requirements for the data architecture based on the project's needs are identified. After that, an analysis of existing approaches for data architectures in the industry is done to motivate the DENiM architectural approach. This is followed by the architectural technical specification, and then the platform services to support the tools and user interaction are identified.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2 Semantic modelling

This chapter of the document is dedicated to the use of semantic modelling within the DENiM project. First, a general introduction to semantic modelling is given in Section 2.1, followed by an analysis of the semantic requirements in the DENiM project (Section 2.2). Section 2.3 presents an analysis of existing approaches, concepts and technologies regarding semantics (both in a general context and specific to the energy and manufacturing sector). The selection of concepts and technologies included in this subsection is motivated by the identified requirements from the previous subsection. Section 2.4 introduces the DENiM semantic data models and ontologies semantic approach that will be adopted for the project.

2.1 Introduction to semantic modelling

A semantic data model (SDM) consists of a conceptual data model that includes semantic information, which means that the model describes the meaning of its instances and the knowledge there is about the subject and the instances. A good semantic design and specification are important in order to manage large amounts of complex data from diverse sources, supporting data usability. With semantic data modelling, developers fully define the meaning of data within the context of its interrelationships with other data. In addition, semantic data modelling enables automated data interoperability in, e.g. agent-based systems. In summary, an SDM is an abstraction that specifies how the symbols stored in a database relate to the real world.

In the World Wide Web Consortium (W3C) *Data Activity* (World Wide Web Consortium (W3C), 2021a), one of the most common semantic data modelling frameworks, data and information are managed in a *graph*-based form, and the foundation for data representation is a *data triple*. A graph is a data construct in which data is organised into nodes or vertices, and edges or links between nodes. Both nodes and edges can contain information, and edges can be directed, meaning that the direction from one node to another matters. Graphs represent data or information, which is linked and networked, akin to communication networks, systemic models, and knowledge. A data triple consists of two data objects and a connection relation, and they form a simple directed graph of a subject (a data object), a predicate (a relation), and an object (another data object). The semantics, i.e. the meaning of the triple, relies on the types of the graph objects and relations, in the values of attributes, and in the links between the graph objects. The types of objects (classes), their attributes (data properties), and relations between objects (object properties) are defined in ontologies.

Let us consider a class of object, *person*, and relation types (i.e. object properties), *has-a-child*. In a simple data model, based on an ontology, we have two instances of the class *person*:

- Peter (instance of the class *person*)
- Alice (instance of the class *person*)

With the two instances and a relation, we can already represent information with just one data triple (Figure 1).

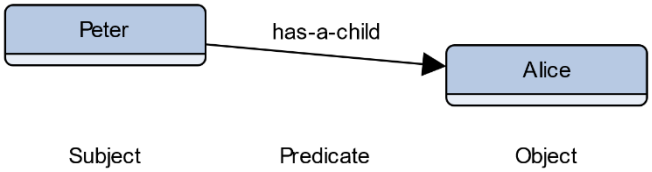

Figure 1: An example of a data triple, modelling the information that Peter (subject) has a child (predicate) Alice (object).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

To have more information in the data model, the classes can have attributes (i.e. data properties), such as *has-age*. The data model can then contain more information (Figure 2).

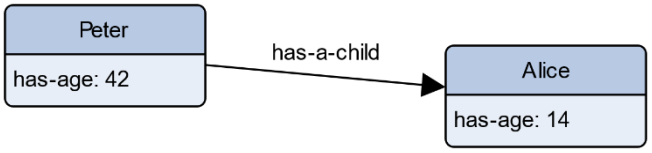

Figure 2: An example of having attributes (data properties) defined for the data objects (class instances) in a semantic data model.

The simple example above shows that information can be modelled with simple elements. The concept of a data triple can form chains and graphs, and the object of one data triple (subject–predicate–object) can be a subject of another data triples.

To extend the capabilities of semantic modelling and to enable knowledge representation, we need to be able to infer new information, or knowledge, from the existing information or data. To do that programmatically, we need formalisation of the data and information representation and formal logics for inferring the data and information. Let us extend the simple example above with a new relation, *has-a-grandchild*, in the ontology, a third instance of class *person*, Richard; has-age: 71, and a relation with data object *Peter*. Now, we have a graph of two triples and explicit information about three persons, their age, and their relationships (see Figure 3).

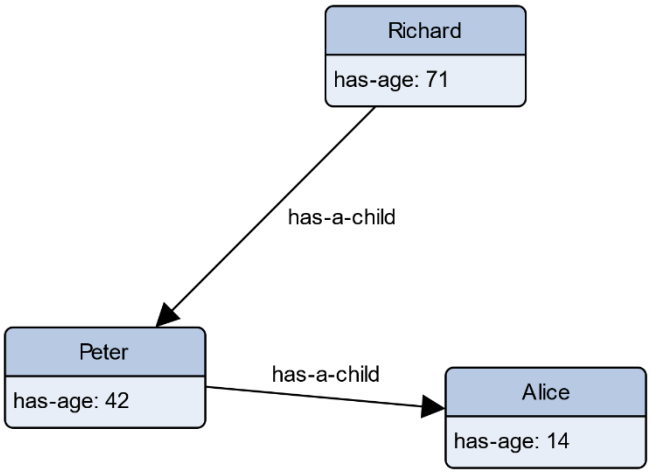

Figure 3: The example data model in Figure 2 extended with a new object and a relation. The graph has two data triples.

Based on our knowledge and the given information, we can infer that Richard, in addition to being the father of Peter, is also the grandfather of Alice. To infer this programmatically, we need to have logical rules, e.g.:

IF (a : *person*) <has-a-child> (b : *person*)

AND

IF (b : *person*) <has-a-child> (c : *person*)

THEN (a : *person*) <has-a-grandchild> (c : *person*)

With our simple data model, this logical rule would give that Richard has a grandchild Alice. This information is not explicitly in the data model, but it is inferred based on our prior knowledge about family relations and the data in the graph (Figure 4).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

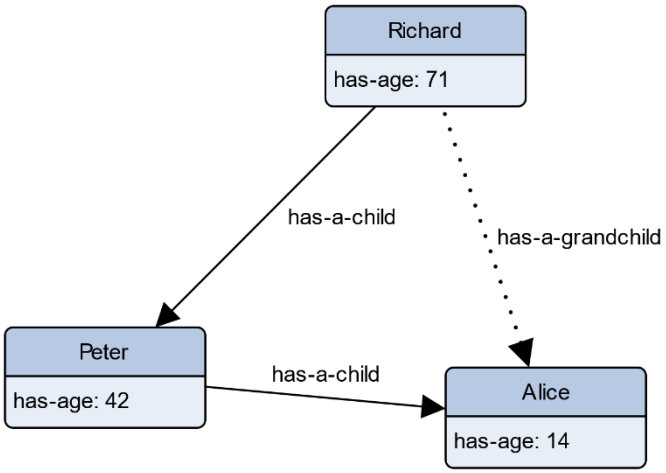

Figure 4: An example of inferred information ("Richard has a grandchild Alice"), based on the prior knowledge of family relations and the explicit information in the data model.

In knowledge representation and in many of the technologies and standards for the Semantic Web, two fundamental concepts require further discussion, namely the *open-world assumption* (OWA) and *unique name assumption* (UNA). When representing knowledge, in principle, no one can possess all the knowledge about a *thing*, or it cannot be guaranteed. The world is assumed to be open, and there can be someone who has additional knowledge about a *thing* (the OWA concept). It is just not known by that observer. This means that, for example, listing all the (obvious) possible attributes of a *thing* cannot be used for validating that the *thing* is fully defined and that there cannot be more attributes. This is a fundamental feature and must be considered if semantic modelling technologies are to be used for, e.g. defining data models and for programmatically validating the data. A similar kind of concept is the UNA. It means that a *thing* can have only one and unique name and/or identifier, which fully identifies the *thing*. Most of the Semantic Web technologies do not make this assumption and allow the same *thing* to have multiple names and identifiers. If multiple instances of a *thing* used in knowledge representation refer to the same *thing*, it can be explicitly defined. These concepts define the foundation for logical representation of knowledge and programmatic reasoning and inferring in semantic modelling.

2.2 Semantic requirements in DENiM

In this section, a brief analysis of the general requirements in terms of semantics for the DENiM project is presented. The main objective of this chapter is to expose the main drivers for the semantics strategy to be followed within the project.

One of the main requirements of the DENiM project is semantic interoperability, which is the ability to exchange data with unambiguous meaning. DENiM aims to develop technologies and produce outcomes whose data can be shared across the four pilot use cases and ideally beyond them in order to support replicability. This poses strict requirements on using a common language for the produced data to be seamlessly interchangeable. This common language is defined as an ontology or a set of ontologies. Being able to share data, together with their meaning and unambiguously, demands the use of a common, defined ontology.

However, the set definition of ontologies to be used in the project would imply introducing restrictions in the scope and usability of the developed tools. The DENiM data platform is being defined as a reference architecture and designed without focusing on a single industrial sector. As such, the DENiM platform is an industry-agnostic set of technologies and services for the acquisition, ingestion, management, storage and serving of both real-time and historical data of any type. But for this set of

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

tools to be usable, the user needs to be able to customize the platform for their specific use case, and this is achieved through the use of semantics. This objective can be achieved through the definition and customisation of semantic models by the user to match specific use case needs. This model will be subsequently used in setting the appropriate configuration of the platform (e.g. data structures, data topics, end points etc) to match those semantics needs. The definition of semantic data models will be done by using a semantic editor (like, for instance, Protégé) to create instances of the ontology classes. For example, a user could define ten machines, with one sensor each, and the measurements that each sensor produces, together with their units. If, at some point, the template ontology is not enough for the user needs (imagine needing to define a value for the precision of the sensor and not having that possibility with the default template), the base ontology can be enhanced by creating new classes or modifying the existing ones.

At the same time, the custom semantic models defined by the use case developers need to be linked in some way to avoid different definitions of the same concepts, which would lead to incompatibilities and interoperability issues. For this reason, common source ontologies covering the identified needs will be defined so that the developers/future users of the DENiM platform can create customized ontologies inheriting from those root ontologies.

The main ontology to be designed as part of the DENiM platform is based on the primary type of data that is to be managed by the platform in the DENiM project – this is energy-related data in the manufacturing field. However, the set of ontologies to be created for the DENiM project also need to cover the different tools and components, processes or productions that will either produce or consume data as part of the energy management process. This includes the following DENiM specific elements, which correspond to the functional blocks defined as part of the DENiM requirements, with the exception of those that are internal to the data platform:

- Machine Learning at the Edge
- Energy Twins and Digital Twins
- Online Lifecycle Analysis (LCA) and Lifecycle Cost Analysis (LCCA)
- Energy Performance Indicators
- Sustainable Production Planning
- Model Predictive Control for Renewables integration
- Fault Detection and Diagnosis
- Context-Driven Visualisation of Performance Metrics
- Digital Readiness Indicator
- Standards-based Energy Auditing

To identify the needs regarding the semantics of the components listed above, a component specification document was created to identify the input and output data requirements for each of the proposed DENiM tools. Analysis of the component specification documents will allow the platform developers to select a set of existing industry ontologies that best fits the project's needs. This use of existing ontology definitions supports standardisation (use of and potential extension) and can be considered as a template that can potentially be extended with more definitions as needed.

An initial analysis of the general ontologies and semantics tools relevant to DENiM is presented next, and the DENiM semantic approach will then be described in Section 2.4.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2.3 Existing approaches, concepts and technologies

The origins of data semantics are in knowledge engineering (KE) and early work in artificial intelligence (AI). The concept of data semantics can be approached from the different levels of information. The technologies developed for semantic data management and especially for linked data can be used for managing and processing low-level data, especially if the data is linked and naturally graph-forming. An example of naturally graph-formed data is the description of an information technology network consisting of servers and computers (nodes) and data connections (edges). Semantic technologies are better suited for information and knowledge representation and analysis. For knowledge engineering, the ability to infer formal data and to validate the data against given definitions opens interesting possibilities and enables automated processes for complex data management.

2.3.1 Semantic Web technologies

The W3C Semantics Web standards stack (Figure 5) is the most widely used for semantic data management. The aim of the development of the Semantic Web is to extend the World Wide Web by bringing "*structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users*" (Berners-Lee *et al.*, 2001). From the outset, Semantic Web technologies were developed to be general and to be used for purposes in addition to the World Wide Web - e.g. the concept of a *resource* refers to any resource, either virtual or physical, that can be accessed or referred to.

The stack of Semantic Web standards and technologies builds on top of lower-level technologies, such as Extensible Markup Language (XML) (World Wide Web Consortium (W3C), 2008), (World Wide Web Consortium (W3C), 2006) and Unicode and contains the required building blocks to describe and link data, to query data, and to include necessary constraints and rules for information reasoning and inferring. The W3C Semantic Web standards are listed, together with a large set of other W3C standards, in *W3C All Standards and Drafts* web page (https://www.w3.org/TR/).

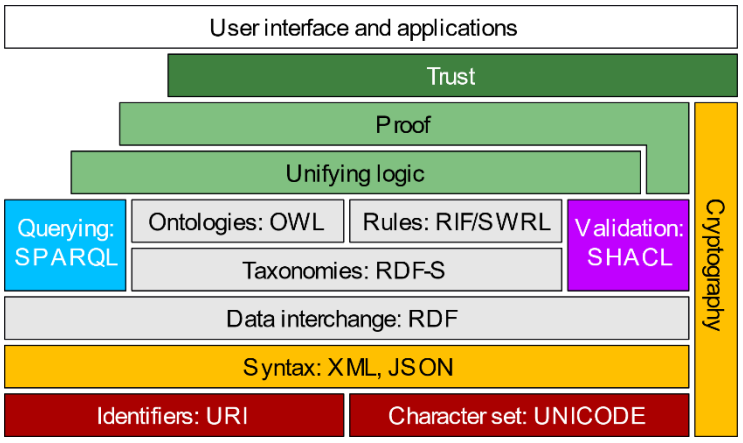

Figure 5: The Semantic Web stack (based on https://en.wikipedia.org/wiki/Semantic_Web_Stack, Creative Commons CC0 1.0, public domain).

The work that started as the Semantic Web has been extended to cover other use cases for the semantic, graph-based data and its applications beyond the World Wide Web, and is nowadays called W3C Data Activity, which clearly indicates that the use of these technologies has extended from Web browsing to machine-to-machine applications and the use of semantically enriched data for automated data, information, and knowledge processing.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2.3.1.1 Resource Description Framework, RDF

The Resource Description Framework (RDF) forms the basis of the Semantic Web technologies (World Wide Web Consortium (W3C), 2014a, 2014b, 2014c). The main concepts defined in the RDF are the graph-based data model and the data triple (see Section 2.1). In addition, the Standard defines the main elements in relation to resource descriptions, starting with the definition of a resource and denoting that, in this context, anything can be a resource. In a data graph, resources are either nodes or literals, i.e. data values with a type. Nodes are identified using Internationalized Resource Identifiers (IRIs). The RDF Standard defines the semantics and syntax of the main elements and defines concepts, such as the RDF vocabulary, RDF datasets and datatypes. RDF lays the basis for linked data and its fundamental concepts. It is a relatively low-level description and does not provide necessary definitions for proper knowledge representation and for programmatic reasoning and inferring.

2.3.1.2 RDF Schema, RDF-S

The RDF Schema (RDF-S) (*RDF Schema 1.1*, 2014) adds definitions and constraints on top of RDF and is one of the fundamental RDF vocabularies. It also provides the building blocks for other Semantic Web standards and technologies, such as the Web Ontology Language (OWL). RDF-S defines the concepts of a *class* and a *property*, which are used and further defined in OWL for knowledge representation.

2.3.1.3 Web Ontology Language, OWL

The Web Ontology Language (OWL) (World Wide Web Consortium (W3C), 2012b, 2012c, 2012a) is a semantic language for information and knowledge representation. The language further builds on top of the basis of RDF and RDF-S by defining more concepts, restrictions, constraints, and principles. These are fundamentally needed to enable programmatic reasoning and inferring of information and knowledge represented with OWL. An ontology is a set of related definitions of concepts (classes), their attributes (data properties), and their relations (object properties) that define the vocabulary and principles of a dedicated domain of interest. In addition to defining the main concepts, OWL also defines the first-order logics that enable reasoning. OWL makes the open-world assumption (OWA) but does not make the unique name assumption (UNA). These are important to notice when considering the use of OWL for data modelling and data validation.

2.3.1.4 SPARQL Query Language, SPARQL

The data in RDF and OWL is organised in the form of graphs and requires dedicated means for querying. The SPARQL Query Language (SPARQL) (World Wide Web Consortium (W3C), 2013c, 2013d) is a standardised language for data querying and manipulation in the Semantic Web context. SPARQL provides a roughly similar kind of functionality as Structured Query Language (SQL) does for relational databases. This language is one of the foundational components of Semantic Web technologies.

2.3.1.5 Semantic Web Rule Language, SWRL, and Rule Interchange Format, RIF

RDF, RDF-S, and OWL lack the means to define rules and constraints as part of ontologies and data models. There are many rule languages that are used for adding restrictions to data models, such as the Semantic Web Rule Language (SWRL) (World Wide Web Consortium (W3C), 2004a). In addition, there is a dedicated language for exchanging rules among different systems, the Rule Interchange Format (RIF) (World Wide Web Consortium (W3C), 2013a), (World Wide Web Consortium (W3C), 2013b). It should be noted that SWRL has not gained the status of a W3C recommendation, but it is relatively widely supported.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2.3.1.6 Shapes Constraint Language, SHACL

The Shapes Constraint Language (SHACL) (World Wide Web Consortium (W3C), 2017b), (World Wide Web Consortium (W3C), 2017a) is the latest addition in the W3C standards for the Semantic Web. It is a language for describing and validating RDF graphs, i.e. it is a language for describing and validating data represented in RDF graphs. It is closer to defining XML Schema (World Wide Web Consortium (W3C), 2004b), (World Wide Web Consortium (W3C), 2012d), (World Wide Web Consortium (W3C), 2012e) constraints that to defining rules and constraints for knowledge representation in OWL. The SHACL shapes can define and validate, e.g. the types of data nodes, their cardinality, and data types of property values. It also enables to define the allowed properties for node instances, which means it has closed world assumption features, which, on the other hand, makes it more suitable for data validation purposes. As being a rather new standard, the availability of tools and systems supporting SHACL is still limited.

2.3.1.7 Extensible Markup Language, JSON and related technologies

Other supporting technologies, such as XML and Extensible Stylesheet Language Transformations (XSLT) (World Wide Web Consortium (W3C), 2017c), exist for, e.g. data serialisation and for transforming data between different serialisation representations. These may be needed both for the data exchange of data models as well as for the actual data. The W3C standards define serialisation of, e.g. RDF and OWL model to several formats, such as N-Triples, Turtle, RDF/XML, TriG, N-Quads, and JSON-LD.

2.3.1.8 Tools and platforms for linked data, Semantic Web, and W3C Data Activity

The W3C Semantics Web stack and Data Activity standards and technologies, together with the supporting technologies, lay the basis for lower-level data management as well as for semantic reasoning and formal knowledge management. Yet, for the real-world use cases, the implementation of the standards and technologies in the form of software tools, libraries, platforms, and frameworks is needed. The validation of RDF formed data by using SHACL requires the data graphs of the data to be validated, the SHACL shapes graph containing the validation rules, and the validation engine to perform the validation. To create the SHACL shapes graph, an editor is needed, which can be a regular text editor, but for better productivity, it would merely be a dedicated editor tool for the purpose. Or, for using semantic reasoning, again, the target data graph, the related ontologies, and a reasoner software are needed.

2.3.1.8.1 Protègè

Protègè is a free, open-source, and widely used OWL editor and framework for editing ontologies, creating data models based on the ontologies, querying data models, reasoning, and visualising the ontologies and data models, among other things. The editor is implemented using the Java programming language, and it has a large set of available plugins that extend the editor's basic capabilities. There is also a Web-based (browser) version, WebProtègè, of the editor available with slightly limited capabilities compared to the standalone Java version. The development of the Protègè editor has been slow during the past number of years. The latest version, Protègè 5.5.0, was released on 15 March 2019. (University Standford, 2021)

2.3.1.8.2 TopBraid Composer

The TopBraid Composer by TopQuadrant is a commercial integrated development environment (IDE) for semantic graph and linked data services. The IDE is built on the Eclipse platform and has tools for, e.g. OWL ontology and RDF vocabulary development, validation, and reasoning. (TopQuadrant, 2021)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2.3.1.8.3 RDF4J

The RDF4J Framework, based on the Eclipse platform, is targeted for RDF-based graph data and for software development of graph and linked data solutions. The platform is a Java framework that provides ready-made components and application programming interfaces (APIs) for software application development. The platform architecture is presented in Figure 6. (Eclipse Foundation, 2021b)

The RDF4J framework contains implementations of three databases for graph data (Eclipse Foundation, 2021b):

- The RDF4J Memory Store is an in-memory database for fast access to a relatively small amount of data. The database supports persistent synchronisation to disk.
- The RDF4J Native Store is a database that uses direct persistent I/O to disk. This database is more scalable than the Memory Store database and is aimed at for medium size datasets.
- The RDF4J ElasticsearchStore is an experimental database based on the Elasticsearch search and analytics engine (Elasticsearch B.V., 2021)

In addition, there are several third-party RDF databases that utilise the RDF4J framework APIs, such as Ontotext GraphDB, Halyard, Stardog, and Amazon Neptune. The RDF4J framework is licensed under the Eclipse Distribution License (EDL), version 1.0. (Eclipse Foundation, 2021b)

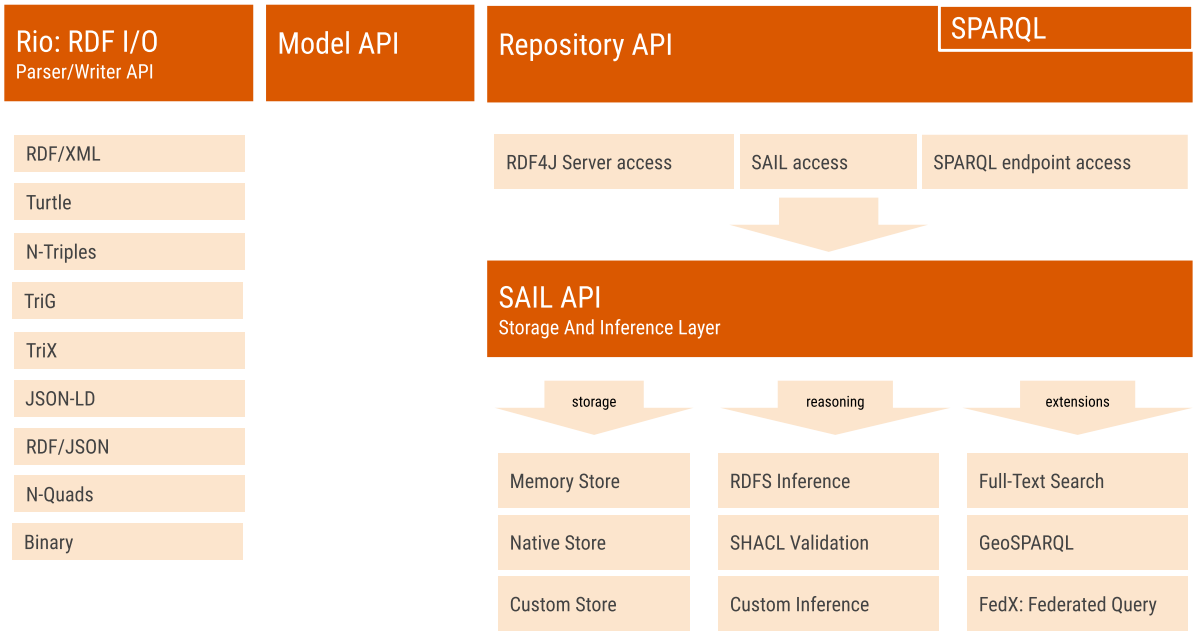

Figure 6: Eclipse RDF4J platform architecture (redrawn from https://rdf4j.org/about/).

2.3.1.8.4 Apache Jena

Apache Jena is an open-source Java framework for software development of the Semantic Web and linked data applications. The RDF4J and Jena frameworks share a similar concept, both being Java platforms for application development and providing similar kinds of interfaces and APIs. In addition to providing support for RDF data, Apache Jena also directly supports OWL. The framework has long development history, and its origins were in the HP Labs in 2000. Nowadays, it is part of the Apache Foundation's software distribution and licensed under the Apache License, version 2.0. The architecture of the Apache Jena platform is presented in Figure 7. (Apache Software Foundation, 2021)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

The Apache Jena framework includes an RDF database, TDB, for data storage and query. In addition, the framework has a SPARQL end-point for RDF data and several APIs for direct SPARQL queries, reasoning and inferring, and working with semantic OWL data. (Apache Software Foundation, 2021)

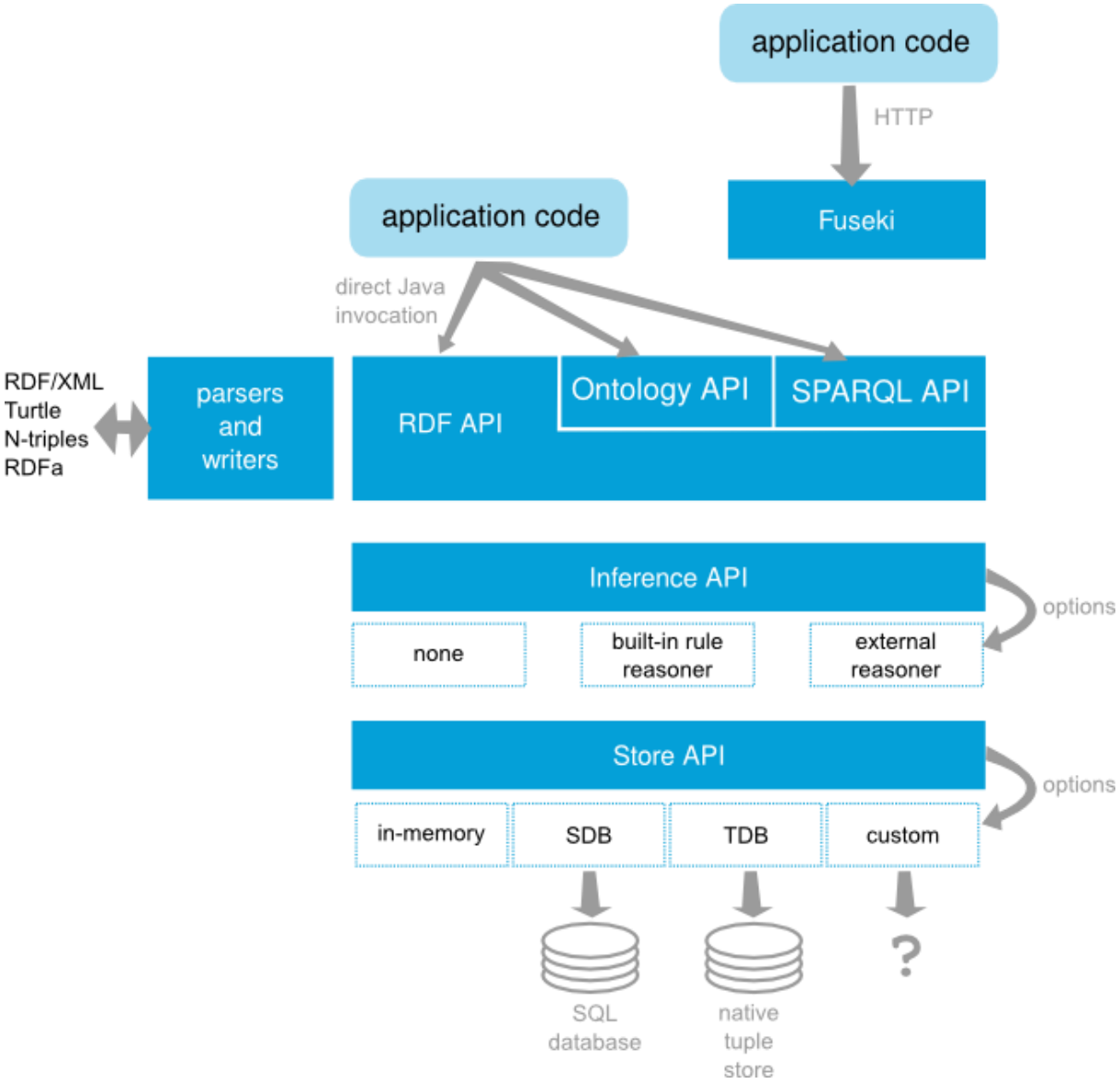

Figure 7: Apache Jena framework architecture (source: https://jena.apache.org/getting_started/index.html, Apache License, version 2.0).

2.3.1.8.5 Eclipse KOMMA

Eclipse KOMMA is an RDF object mapper and editing framework for Java, and it is based on the Eclipse RDF4J framework. The framework simplifies the software development by providing an improved Java method interface for graph data and provides an editor for RDF, RDF-S and OWL data. (Fraunhofer IWU., 2021)

2.3.1.9 Other relevant concepts: Web of Things, WoT and Asset Administration Shell

W3C has started an activity, Web of Things (WoT), to standardise the main technologies of the Internet of Things (IoT). The aim of the activity is to complement existing standardisation and to improve

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

interoperability rather than to develop new technologies. The WoT defines few fundamental concepts for IoT, such as (World Wide Web Consortium (W3C), 2021b):

- WoT Interactions, a simple interaction abstraction model of the interface that is based on properties, events, and actions;
- WoT Thing Description (TD), metadata description for the IoT device on which data and functions are provided, which protocol is used, how data is encoded and structured, and what security mechanism is used to control access;
- WoT Binding Templates, description of communication protocol mappings to enable consistent communication of WoT elements using different protocols, such as MQTT, HTTP, CoAP or Modbus;
- WoT Scripting API, an optional Javascript API for interaction;
- WoT Security and Privacy guidelines for implementing WoT systems.

The currently available W3C documentation on WoT can be found at W3C all standards and drafts web page (https://www.w3.org/TR/?title=wot).

Another interesting concept related to information exchange is Asset Administration Shell (AAS), a concept created by ZVEI, the German Electrical and Electronic Manufacturers' Association. AAS is used to describe any asset electronically in a standardised manner to exchange asset-related data among industrial assets and between assets and production orchestration systems or engineering tools (de Leeuw, 2019). One fundamental difference between AAS and other existing solutions is that AAS provides manufacturer-independent standardisation of the management shell meta-model. In the field of production automation, there is no current standard for the self-description of systems and devices in a technology-neutral way. An AAS provides a unique identifier for each asset, information about i4.0 assets, and generic interfaces for information exchange. One interesting feature in our field of interest is that AAS has the possibility to link the assets with their semantic specification, containing the meaning of the element. This connection is made through a semantic ID, which is usually a reference to an external global ID or to a model element containing the semantics.

There are several open-source initiatives working towards the creation of tools for developers, like admin-shell (https://github.com/admin-shell-io), BaSyx21(https://www.eclipse.org/basyx/) and SAP i40-aas (https://github.com/SAP/i40-aas). However, it must be noted that they are versions under development and mostly of an experimental nature.

The AAS is still being developed, but the authors are putting a big emphasis on converting it into a standard. Currently, existing IEC and ISO standards are considered for use within the AAS for specifying properties and property values. For that reason, several specification documents have already been released (Plattform Industrie 4.0, 2019) (Plattform Industrie 4.0, 2019, 2020).

Many more related technologies and concepts could be included in this report. However, we decided to limit it to the ones that are more extensively used in the industry. Before moving to the section dedicated to energy and manufacturing, we would like to include a reference to OPC UA (Open Platform Communications Unified Architecture), which is a set of communication standards dedicated to digitalization, independent of platform and manufacturer and with integrated security mechanisms. Within its set of features, OPC UA includes information models and semantic services. However, there is a lack of OPC UA formal semantics, and that is why several authors have proposed formal translations to the OWL standard in order to enable automatic validation of OPC UA data models and the seamless usage of OWL query engines.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

2.3.2 Energy-focused semantic models and ontologies for manufacturing

The adoption of energy-saving and environmentally-friendly measurements in the manufacturing industry requires a broad knowledge of manufacturing processes. Available enterprise data can make it possible to derive energy efficiency and material consumption measurements. However, this data first needs to be structured in order to produce meaningful information.

Nowadays, information in manufacturing companies comes from heterogenous sources that usually do not understand each other and have different formats, units, etc. This leads to the need for a semantically rich information model for data sharing and storing.

Traditionally, information models are built with technologies such as EXPRESS (International Organization for Standardization (ISO), 2004), XML or Unified Modelling Language (UML) (Object Management Group, 2017), which do not include semantic content. Enhancing these standards with the use of ontologies is a possible way of achieving representative information models. Another approach to this topic that some companies already use is the creation of RDF databases, where semantic queries can be performed.

Some research works have already proposed information models or methodologies for manufacturing processes, but not much work has been done towards integrating sustainability data. Below, a brief review of some related research projects is presented.

- In (Li, Zhang and Gao, 2010), a model for manufacturing processes is presented. It includes basic information and views for the different applications: process planning, management, virtual reality and resource usage. Semantics are mentioned as a way to describe the design intent.
- (Kim, Manley and Yang, 2006) propose the Assembly Relation Model (ARM), an assembly-focused information model enhanced with ontologies that promote collaborative information-sharing environments. Semantic queries are available for process designers to access the assembly information in collaborative design tools. The project uses an assembly design (AsD) ontology that is machine-interpretable and thus allows knowledge to be automatically managed and shared.
- The Open Assembly Model (OAM) proposed in (Rachuri *et al.*, 2006) is a model that represents assembly processes at a system level and with the associated hierarchical relationships. The focus in this project is set on tolerance, kinematics and engineering analysis, but energy matters are not taken into account.
- The OntoMAS framework from (Lohse, 2006) is a design framework for manufacturing processes that includes an ontology with an extensive set of entities and corresponding relations between features and tasks. Again, sustainability entities have not been considered in this project.
- MASON (MAnufacturing's Semantics ONtology) is an ontology presented in (Lemaignan *et al.*, 2006), based on OWL formal ontology language. The proposal is a very high-level definition of entities and relationships that unfortunately do not take energy concepts into account.
- (Feng, Kumaraguru and Sun, 2015) propose an energy assessment methodology for evaluating efficiency in manufacturing processes. The methodology includes a set of activities and iterative processes to be integrated into the business strategy. However, there is no intention to integrate information from multiple sources or knowledge domains. The project proposes a non-data-analytics-supported decision-making methodology.

From the short review presented above, we can infer a clear gap between the modelling of manufacturing processes and the formal structuring and enrichment of that data to extract energy

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

efficiency-related information that support the transition to sustainability. Energy-focused ontologies and information models are not widely standardised. Some researchers have recently tried to make proposals in this direction.

- (Borsato, 2017) proposed an energy efficiency-focused semantic model for manufacturing mainly focused on assembly processes and showed that it could be extended for many different manufacturing processes. The proposed ontology is then used as a basis for an exergy-based thermodynamic analysis that can calculate the degree of perfection of the process in terms of sustainability.
- (Wenzel *et al.*, 2011) presented their research on semantic web-based energy analysis and forecasts in manufacturing. This work starts from existing ontologies like MASON, Semantic Sensors Network (SSN) or Measurements Units Ontology (MUO) to construct a knowledge base and information model that allows the collection of processes data. Then, that data can be easily analysed and further used for energy efficiency purposes.

We can conclude that while several research projects are making progress in using semantics for energy efficiency purposes in manufacturing, there is no standardised ontology or semantic model dedicated to this. Several energy ontologies exist in other sectors where intelligent energy management is undergoing fast development, like those in the Smart Grid sector (ThinkHome, BOnSAI ProSGV3). However, these ontologies focus on buildings' energy, with many instances dedicated to appliances, wearable devices, etc. Existing general ontologies can be leveraged and enhanced with energy-specific semantic information to achieve a semantic model specific to DENiM. Based on the literature SAREF is a likely candidate as it is the most generic and flexible from the candidates considered, and extensions can be used for the adaption to multiple domains from the existing ontologies. Thus, SAREF will be further analysed in the following subsection.

2.3.2.1 SAREF

The Smart Applications REFerence (SAREF) ontology is a technical specification by European Telecommunications Standards Institute (ETSI) (ETSI, 2021a), (ETSI, 2020). The ontology is intended to enable interoperability for IoT solutions from different vendors. It defines the main concepts for the domain and enables the use of unified terminology in communication between humans and in machine-to-machine communication. The set of SAREF ontologies contains the core ontology and several domain-specific ontologies, called extensions for domains.

The set of SAREF ontologies is designed to be modular and extensible. There are currently ten domain extension ontologies, of which some are still under development:

- SAREF: the core Smart Applications REFerence ontology
- Ontology patterns:
    - SAREF4SYST: ontology pattern for Systems, Connections, and Connection Points
- Extensions for domains:
    - SAREF4ENER: SAREF extension for the Energy domain
    - SAREF4ENVI: SAREF extension for the Environment domain
    - SAREF4BLDG: SAREF extension for the Building domain
    - SAREF4CITY: SAREF extension for the Smart Cities domain
    - SAREF4INMA: SAREF extension for the Industry and Manufacturing domains
    - SAREF4AGRI: SAREF extension for the Smart Agriculture and Food Chain domains

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- SAREF4AUTO: SAREF extension for the Automotive domain
- SAREF4EHAW: SAREF extension for the eHealth/Ageing-well domain
- SAREF4WEAR: SAREF extension for the Wearables domain
- SAREF4WATR: SAREF extension for the Water domain

The SAREF ontologies are well documented at the SAREF website[1] , and the source code for the ontologies in OWL/Tuttle format are available at the SAREF Forge repository (ETSI, 2021b). The SAREF ontologies are available as open-source under the 3-Clause BSD License[2].

2.3.2.1.1 The SAREF core ontology

The SAREF core ontology defines the following main classes (concepts) for the smart applications domain:

- **Command**
  - Close command
  - Get command
    - Get current meter value command
    - Get meter data command
    - Get meter history command
    - Get sensing data command
  - Notify command
  - Off command
  - On command
  - Open command
  - Pause command
  - Set level command
  - Set absolute level command
  - Set relative level command
  - Start command
  - Step down command
  - Step up command
  - Stop command
  - Toggle command
- **Commodity**
  - Coal
  - Electricity
  - Gas
  - Water
- **Device**
  - Actuator
    - Switch
      - Door switch
      - Light switch

(Continues from the previous column)

- **Function**
  - Actuating function
    - Level control function
    - On-off function
    - Open close function
    - Start-stop function
  - Event function
  - Metering function
  - Sensing function
- **Measurement**
- **Profile**
- **Property**
  - Energy
  - Humidity
  - Light
  - Motion
  - Occupancy
  - Power
  - Pressure
  - Price
  - Smoke
  - Temperature
- **Service**
  - Switch on service
- **State**
  - Multi level state
  - On off state
    - Off state
    - On state
  - Open close state
    - Close state
    - Open state
  - Start stop state

[1] The SAREF ontology website: https://saref.etsi.org/

[2] The 3-Clause BSD License: https://opensource.org/licenses/BSD-3-Clause

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- Appliance
- HVAC
- Meter
- Sensor
  - Smoke sensor
  - Temperature sensor
- **Feature of interest**

(Continues in the next column)

  - Start state
  - Stop state
- **Task**
- **Time**
- **Unit of measure**
  - Currency
  - Energy unit
  - Illuminance unit
  - Power unit
  - Pressure unit
  - Temperature unit

The structure of the SAREF Core ontology is visualised in Figure 8.

The meaning of each of the classes (concepts) is defined within the ontology. The concept of a device has been described as (ETSI, 2021a): "A tangible object designed to accomplish a particular task. In order to accomplish this task, the device performs one or more functions. For example, a washing machine is designed to wash (task) and to accomplish this task, it performs a start and stop function." In addition to the description, the ontology contains semantic constraints and restrictions for the classes, object properties (relations) and data properties (class attributes) (e.g. defined object property domains and ranges, and data property types and cardinalities). The definition of the device class in the Protègé ontology editor is shown in Figure 9, including the description of the class, the data properties for the class and their cardinalities.

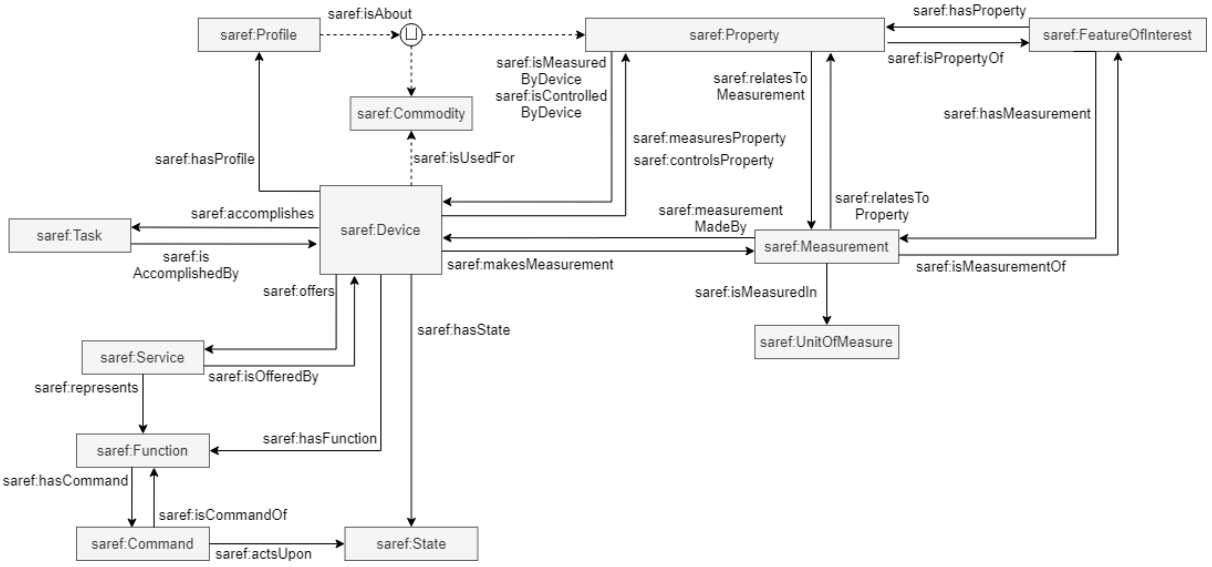

Figure 8: Overview of the SAREF ontology (source: https://saref.etsi.org/core/v3.1.1/).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

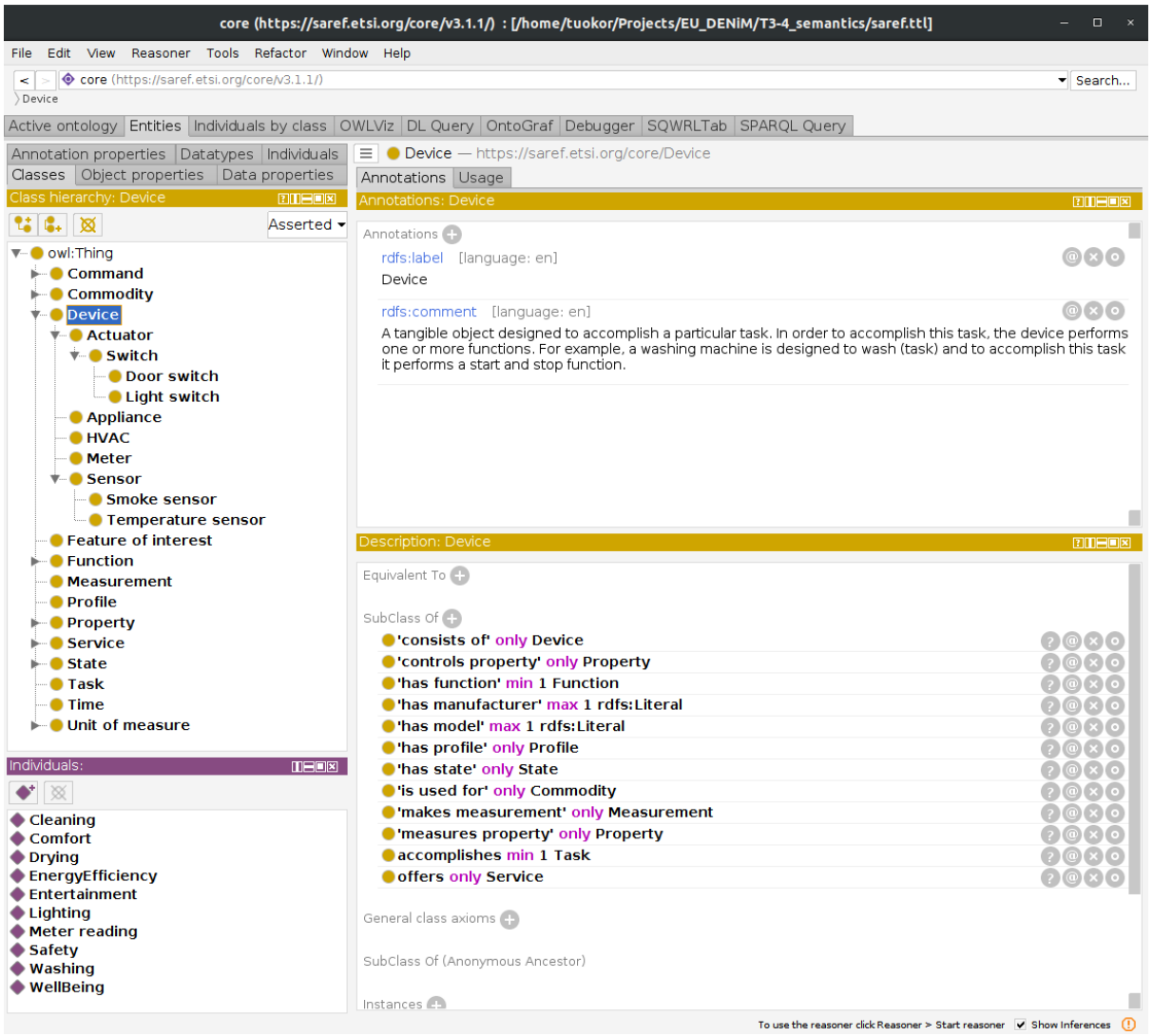

Figure 9: The SAREF Core ontology and the definition of the device class in the Protègè ontology editor.

2.3.2.1.2 SAREF for the industry and manufacturing domain extension

The SAREF extension for the Industry and Manufacturing domains, SAREF4INMA, extends the SAREF core ontology with concepts that are relevant in the manufacturing and production context (ETSI, 2019). The ontology defines concepts (classes), such as factory, site, item, item batch, and production equipment. The structure of the ontology is shown in Figure 10, and an example of the class definition for the concept of production equipment is shown in Figure 11.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

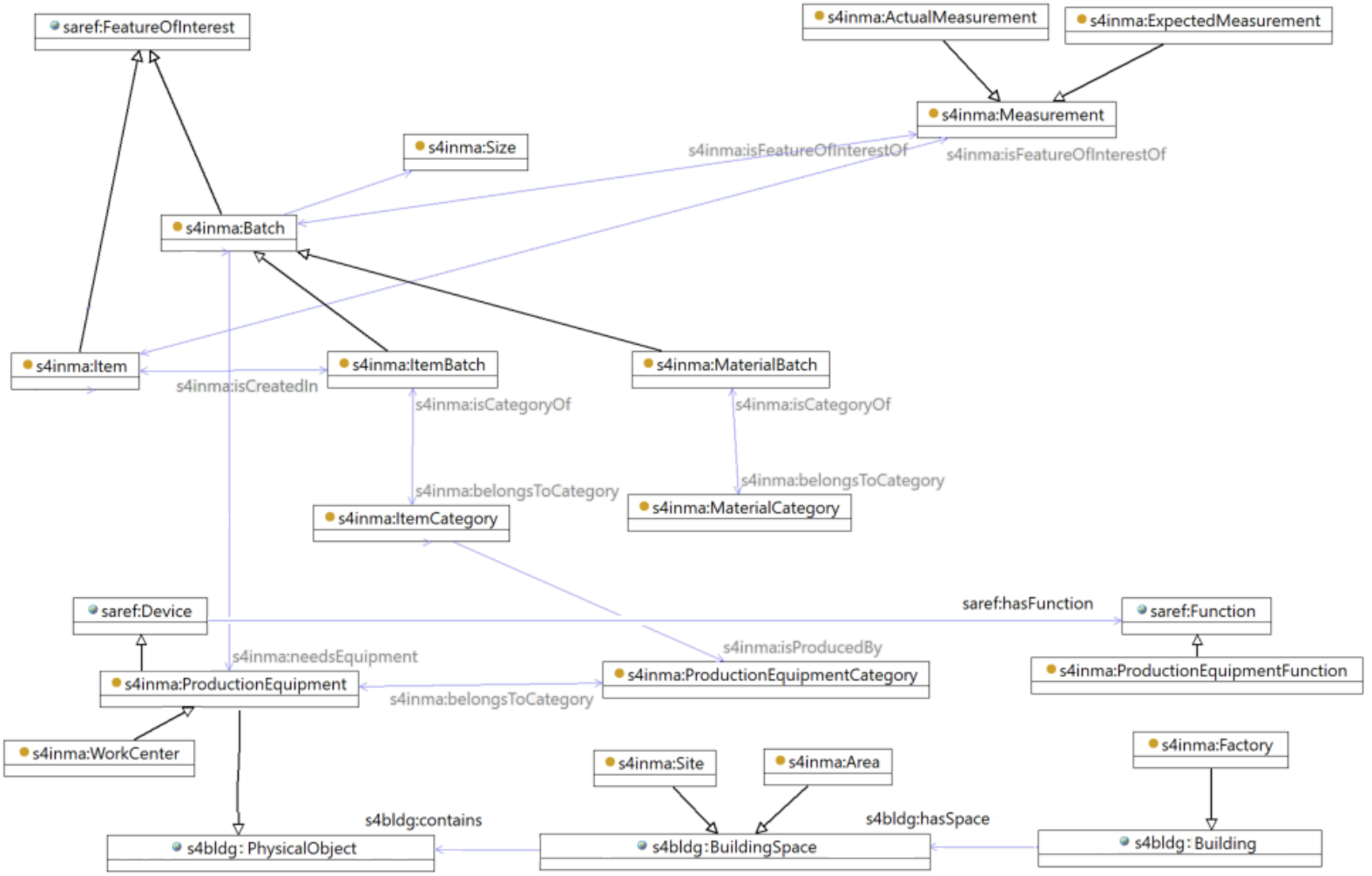

Figure 10: Overview of the SAREF4INMA ontology (source: https://saref.etsi.org/saref4inma/v1.1.2/).

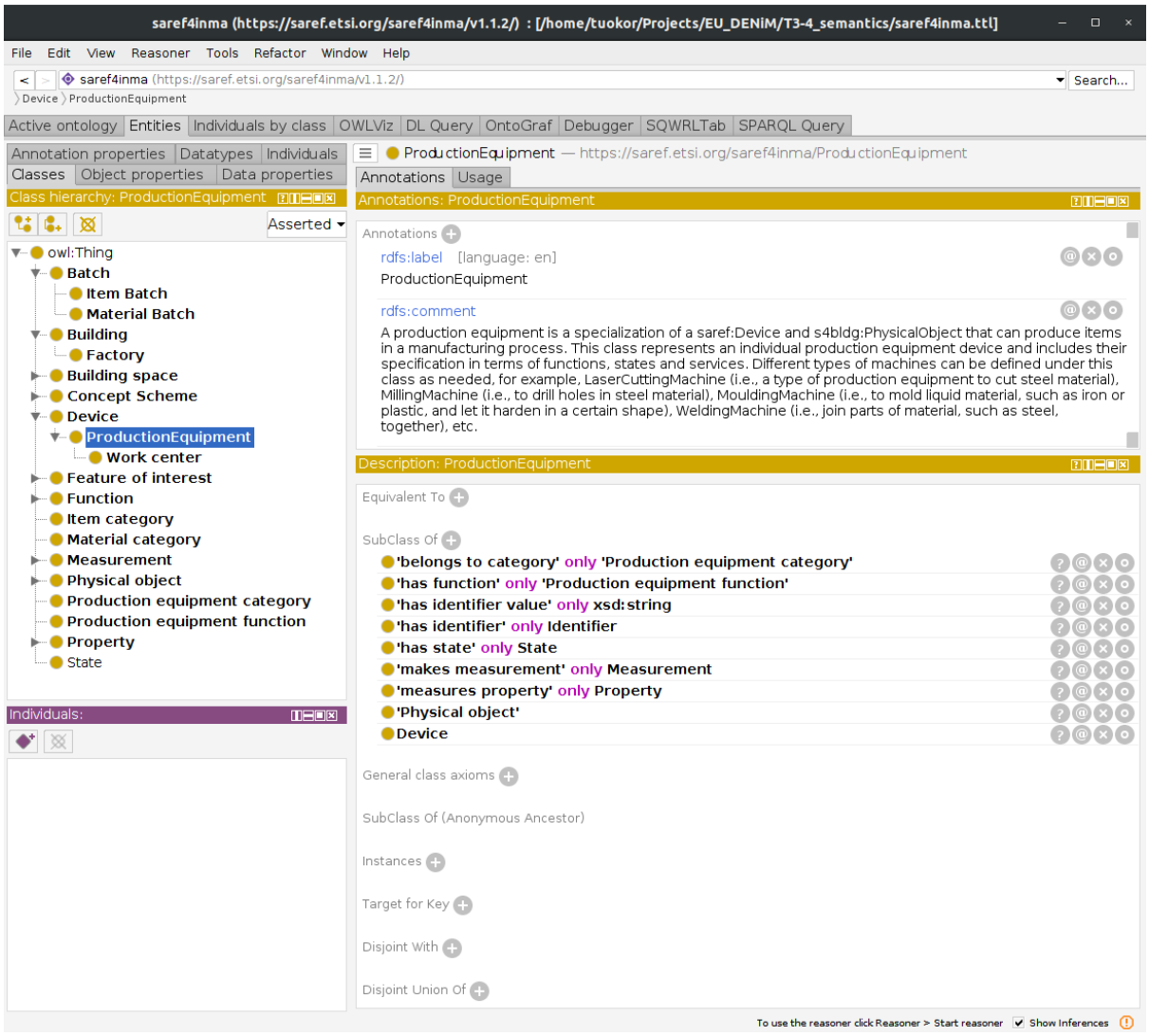

Figure 11: The SAREF4INMA ontology and the definition of the production equipment class in the Protègè ontology editor.

2.3.3 Semantics for data modelling and modelling of systems

Data modelling and meta-modelling are used in software engineering and in systems engineering, especially in model-based systems engineering. In these contexts, the model-based approach is a means to manage design complexity and automate some engineering and design work process phases. This section does not aim for a deep analysis of semantics dedicated to the modelling of systems, as this will be integrated into the services dedicated to that purpose (mainly the model manager and MLOps pipeline manager services, described in subsections 3.3.4.4 and 3.3.4.5). However, it is worth briefly describing the ontologies and technologies for the representation and sharing of modelling data.

2.3.3.1 Model Data Ontologies

With model data ontologies, we refer to semantic data models which are used to represent the data of models used, e.g. for the simulation of energy usage in production processes and manufacturing. The models can be data-based models, such as artificial neural networks (ANNs) or physics-based simulation models (first principles models). The ontologies of these models describe their computational elements, their relations, types and requirements for the inputs and the forms and types of the outputs.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

The use of ontologies for modelling and simulation has several difficulties because existing taxonomies for one type of modelling, with its specific formalisms, are hard to connect with others. If those connections existed, they could be used in query engines, databases, etc., to increase interoperability and reuse of simulation artefacts. The already described Semantic Web technologies could serve as a critical tool to accomplish this objective.

Since the emergence of the Semantic Web, multiple ontologies for specific domains have been developed. In the modelling and simulation sector, we can name some of the initial efforts towards modelling dedicated ontologies, such as the creation of the Discrete-event Modeling Ontology (DeMO) (Miller *et al.*, 2004), the Command and Control Information Exchange Data Model (C2IEDM) for model interoperability (Turnitsa and Tolk, 2005), the use of the Discrete-event Systems Specification (DEVS) formalism as an ontology for models definition (Zeigler, Praehofer and Kim, 2000), the development of the Process Interaction Modeling Ontology for Discrete-event Simulations (PIMODES) (Lee W., 2006), the development of the Component Simulation and Modeling Ontology (COSMO) (Teo and Szabo, 2008) or the more recent creation of the OntoSTEP (Barbau *et al.*, 2012), an OWL-DL (Web Ontology Language Description Logic) version of STEP that allows logic reasoning and inference mechanisms and thus enhancing semantic interoperability.

2.3.3.2 Technologies for model data exchange

In this section, we identify a list of technologies that are traditionally used for model data exchange. These technologies do not necessarily relate to semantics of model data, but are related to it in the sense of pursuing the same objectives of interoperability and information exchange, so we have included them here as a reference.

2.3.3.2.1 Extensive Markup Language

XML, which has been already discussed in Section *Extensible Markup Language, JSON and related technologies* as a very relevant serialisation technology for the exchange of information, can be used as well for models' definition. In addition, it can make use of a schema language, like the XML Schema, for standardisation purposes. However, XML alone is not able to provide semantics at the level of detail required for most of the modelling and simulation needs, that is, interoperability, integration, reusability, or model discovery.

2.3.3.2.2 Unified Modeling Language

UML is a general-purpose graphical system modelling language originally developed for software engineering. The language was developed by the Object Management Group (OMG), and the language specification is publicly available. Together with other related technologies, such as Meta Object Facility (MOF) (Object Management Group, 2019a), the UML approach enables creating smart system modelling tools with automated code generation and model validation mechanisms. The UML language has an extension mechanism, profiles, that enable the creation of domains specific languages. One example of such a domain-specific profile of UML is Systems Modeling Language (SysML) (Object Management Group, 2019b), a dedicated modelling language for systems engineering.

There are many software tools available for UML and SysML modelling, such as Eclipse Papyrus (Eclipse Foundation, 2021a). The tool supports UML version 2.5 and SysML version 1.1, and a number of other UML profiles.

2.3.3.2.3 Functional Mock-up Interface

Functional Mock-up Interface (FMI) (Modelica Association, 2020), a standard for co-simulation, model exchange; used especially in system simulation (https://fmi-standard.org/); Functional Mock-up Unit (FMU) is a co-simulation or model exchange unit (a ZIP file containing the necessary information and

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

computing elements). FMI is a concrete technology to exchange computational simulation components or to connect two simulation software tools and models at run-time (e.g. mechanical system simulation running in one software application and the control system simulation, controlling the mechanical system, running in another).

2.3.3.2.4 Open Neural Network Exchange

Open Neural Network Exchange (ONNX) is an open format to represent machine learning models (AI@Edge Community, 2021). It is a library that makes models based on machine learning interoperable, making it easier for developers to exploit the most suited software tool without being bound to the framework with which the model was developed initially. It provides an extensible graphical computation model, as well as built-in operator definitions and standard data types. ONNX can support a wide range of frameworks, such as PyTorch, TensorFlow, Caffe, scikit-learn, CNTK, etc. Figure 12 illustrates how ONNX relates to the framework used for training and for the deployment of the models.

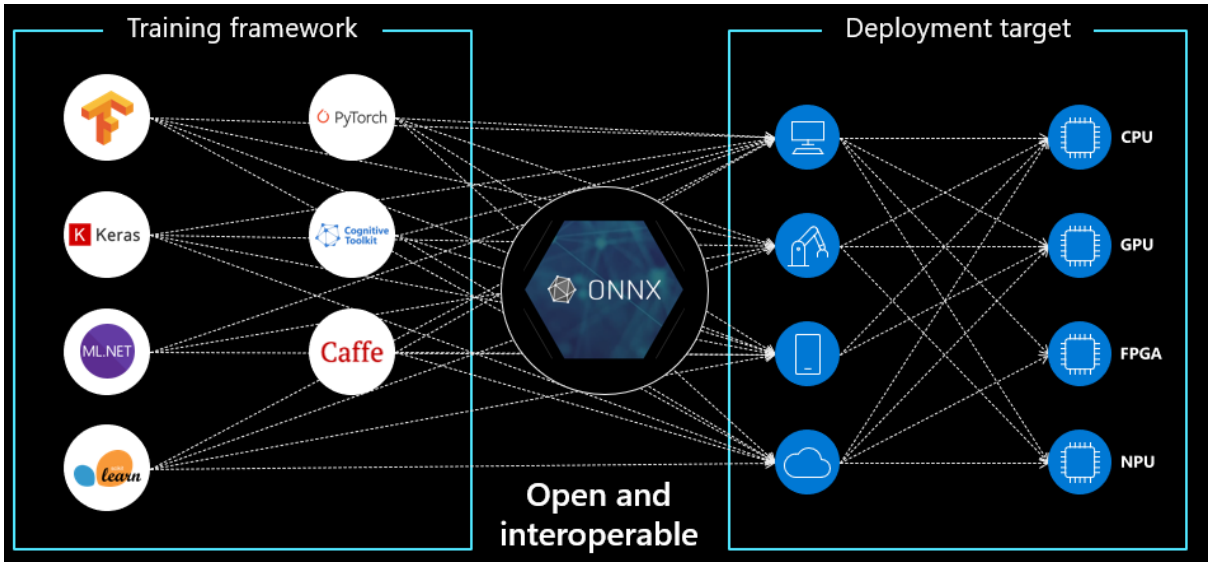

Figure 12: ONNX – Open Neural Network Exchange (The Linux Foundation, 2021).

From the listed tools, ONNX is the most suitable option as it is specifically for machine learning models, and as such, the DENiM platform will support this technology and recommend this format for models involving machine learning. Concerning the other technologies exposed above, we will consider these for different types of modelling if needed. For instance, UML could be used to describe and exchange models that do not involve machine learning techniques.

2.4 DENIM's semantic approach

The requirements identified in Section 2.2 and the existing approaches analysed in Section 2.3 are used to inform DENiM's semantic approach. It is crucial to have a balance between using a standard set of ontologies that enable seamless data sharing and having enough flexibility for developers and users to adapt the ontologies and define their own data models. For this reason, the data platform will be developed in a semantic agnostic way but allowing the integration of multiple semantic standards and custom ontologies.

To use the ontology and the semantic data models, we illustrate three different approaches, each having a different level of complexity and impact on the platform implementation:

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

1) Documentation approach, where the ontology is mainly used for documentation and information exchange between humans. The ontology provides formal documentation of the data flows and enables validation of the data model for the particular application of the DENiM platform.
2) Pre-processing approach, where the ontology is used together with semantic tools to define a skeleton for the platform component implementation for the platform's particular application.
3) Integrated approach, in which the semantic models and the semantic tools are tightly integrated with the DENiM platform and enable automated connection of new data sources, analytics, digital twin modules and user interface elements. This approach is technically demanding.

The DENiM project intends to take a stepwise approach, where the semantic model is initially used for documentation and communication purposes and is progressively integrated within the platform according to the developers and users' needs. The final steps to be followed and the detailed design of the semantic architecture and services will be defined as part of work package 4 activities and reported under deliverable D4.1 DENiM Hybrid Cloud Ready Infrastructure.

Ontologies to be integrated into the platform as default base ontologies for the semantic data models implementation must be based on the fields of knowledge that the developments within the project belong to. These ontologies will be created in collaboration with the developers of the DENiM tools and correspond to the fields identified in Section 2.2. Based on the analysis of existing approaches and standards, SAREF has been identified as the ontology that best fits the context of the DENiM project for energy-related definitions. The following section discusses how the SAREF ontology will be leveraged within the DENiM platform.

2.4.1 SAREF ontology with the DENiM platform

The SAREF ontology provides a well-defined basis for the definition of concepts and improved machine-to-machine communication in the DENiM context. The SAREF core and e.g. SAREF4INMA ontologies can be extended with DENiM specific concepts and definitions. The ambition would be to define a SAREF DENiM extension (named as SAREF4DENiM) which would include classes for data connectors at the DENiM data sources together with detailed data descriptions. In addition, the ontology could include classes for services, digital twin components, data analytics modules, and user interface components, all defining the data connection interface they require and provide.

It is important to remark that, as mentioned previously, it is essential to balance the advantages of the semantic models' usage with the limitations imposed on developers and end-users. Accordingly, the partners involved in the semantics activities of DENiM will support and guide the use case responsible partners, as well as the module developers, in the selection and adoption of ontologies and technologies. However, each developer will have considerable freedom (which might be limited by the needs of the semantic-related decisions done at the module level) to select the most appropriate semantic model they would like to build and which previous developments they would like it to be based on. The proposed approach for semantic modelling corresponds to the need for a common data model identified in the general requirement DI-RQ-04 from D3.1. This will be explored further as part of tool and architecture implementation WP4, WP5 and WP6.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

3 DENiM Digital Intelligence Platform: Architecture specification

3.1 DENiM architecture requirements

Based on the stakeholder and pilot case analysis a set of general DENiM platform requirements were identified as part of DENiM deliverable D3.1. Table 1 presents a summary of those requirements, identifying the key functional blocks required, further details of each requirements is available in D3.1.

Table 1: DENiM platform requirements summary

DENiM Digital Platform: Reliable Data Integration & Sharing	
DI-RQ-01	Digitisation of existing machines and processes
DI-RQ-02	Allow secure integration of device and other systems (data sources)
DI-RQ-03	Supporting binding between devices distributed across different physical networks
DI-RQ-04	Common Data Model to promote data sharing and interoperability
DI-RQ-05	Provide adaptive stream processing
DI-RQ-06	Ensure data integrity
DI-RQ-07	Provide scalable distribution and orchestration of data processing (Hybrid Cloud)
DI-RQ-08	Alignment of the solution with existing standards
DENiM Digital Twin: Accurate Modelling & Verification	
DT-RQ-01	Energy Performance Models (Energy Twins)
DT-RQ-02	Support appropriate selection of approaches for performance modelling
DT-RQ-03	Formalise data integrity strategies to ensure continuous data quality for models
DT-RQ-04	Provide online lifecycle assessment (LCA)
DT-RQ-05	Provide online lifecycle cost assessment (LCCA)
DT-RQ-06	Develop enhanced energy performance indicators (EnPI)
DENiM Decision Support: Decision Support Systems for Sustainability	
DSS-RQ-01	Enhanced energy performance indicators (EnPI)
DSS-RQ-02	Enable production planning that has sustainability as key criteria
DSS-RQ-03	Allow autonomous fault detection and diagnosis
DSS-RQ-04	Enable the integration of onsite renewables with production processes
DSS-RQ-05	Common front-end for performance monitoring
DSS-RQ-06	Assessment of digital maturity of an industrial site

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

DENiM Digital Skills: Digital Skills and Workforce Development	
DS-RQ-01	Effective implementation of standard energy auditing approaches
DS-RQ-02	Identify any skills gap across pilot sectors.
DS-RQ-03	Provide appropriate training to support digital upskilling
DS-RQ-04	Facilitate the sharing of experience and knowledge across representative industry sectors

Requirements are divided according to four corresponding functional groups. The first group is focused on the data management platform which forms the backbone for the other services. The remaining groups concern multiple services and applications that will be deployed and reliant on a common data spine, as they will either consume or produce data via the DENiM data management platform.

3.2 DENiM architecture design

Based on the requirements from D3.1 listed in the previous section, a high level reference architecture design for the DENiM digital intelligence platform is presented in Figure 13. This architecture intends to outline the main building blocks and functional components of the DENiM platform at the current design phase whilst abstracting from the implementation details. This architecture is aligned with the reference architecture from the FoF-09-2017 ForeSee Cluster, dedicated to predictive maintenance for the manufacturing sector (Alexopoulos *et al.*, 2021) and extended to support DENiM specific needs.

The architecture is composed of three vertical tiers, from left to right: 1) the manufacturing site, which contains the physical and virtual components where the DENiM platform is to be applied; 2) the data acquisition and interoperability tier, which contains the link between the manufacturing site and the platform itself and is governed by an edge security layer and the semantic model for data interoperability; and 3) the Digital Intelligence Platform, which implements the data management platform, together with the internal services for the users and applications to interact with it. It addition it captures the external functional blocks for several modelling, control and business intelligence applications included within DENiM. These components are described in more detail in the following subsections.

3.2.1 Manufacturing site

The manufacturing site contains all the elements that are relevant for the implementation and usage of the Digital Intelligence Platform. It is divided into three blocks. First, we find the production systems/value stream and their corresponding virtual assets. This block includes sensors and control systems from where to acquire preferably real-time streaming data and optionally be retrofitted. Second, the enterprise systems (ERP, MES, etc.) may contain relevant information from the production ecosystem perspective, such as production plans, site configuration, energy consumption reports, etc. Finally, the users involved in the production process, management, and platform usage are captured in the users block. Users represent individuals that have expert knowledge about a specific process or product and support optimisation of same through the use of the DENiM solution, for example, operators, engineers and managers, are included in the users' block.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

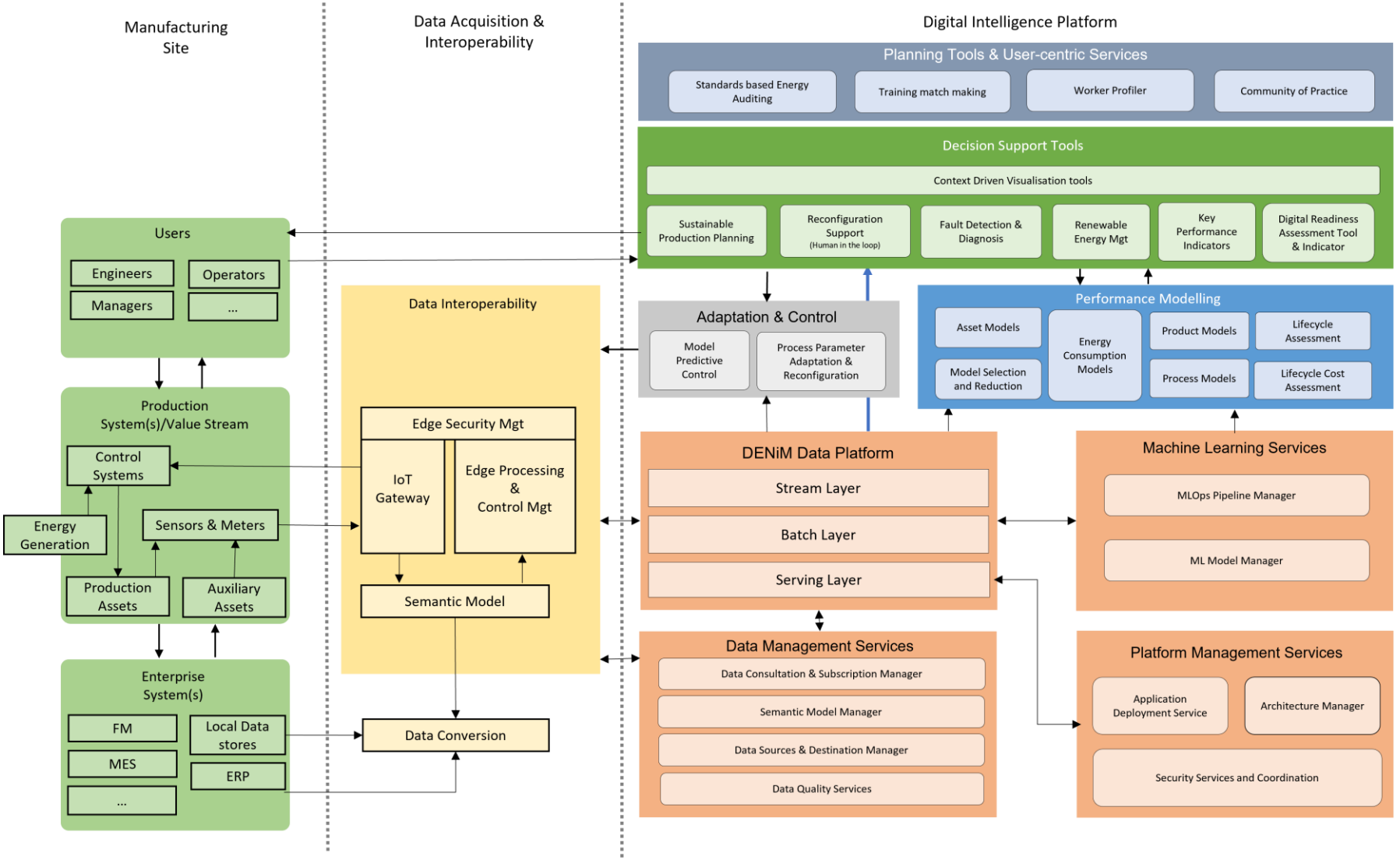

Figure 13: DENiM architecture design

3.2.2 Data acquisition & Interoperability

This tier contains the components dedicated to making both stream and batch information available to the platform. For that, data needs to be appropriately acquired, pre-processed when required and forwarded to the system.

Streaming data from sensors is gathered by the IoT gateway, which makes the sensors available in a common network for the platform to be able to access them. In order to support the acquisition from multiple communication protocols in a secure way, the Secure edge integration and interoperability manager (described under section 3.3.4.9) contains both an edge security management component and an edge pre-processing engine that is able to acquire the data, process it to match the platform semantic model and push it to the platform through a custom connector. This component may also implement preliminary data cleaning, outlier detection or initial data analytics.

Batch data coming from enterprise systems or local data stores needs to be mapped onto the semantic data models before communicating it via the DENiM platform. Following this, it is acquired by the data platform through dedicated historical data acquisition technologies that directly send the information to the batch layer.

3.2.3 Digital Intelligence Platform

This tier contains the main building blocks of the DENiM solution that uses the data coming from the manufacturing site and produces valuable control and business intelligence utilities. These functional blocks can also be grouped depending on the DENiM strategy they belong to, as shown in Table 2. The functional blocks in the table might not fully match the naming in the figure above as the architecture schematic attempts to be a conceptual view of the project, while the table is more dedicated to specific implementation blocks. For that reason, some of the blocks in the figure might correspond to more than one functional block in the table and vice versa.

Table 2: DENiM functional blocks

DENiM Strategy	Functional Group	Functional Block ID	DENiM Functional Block
Reliable Data Integration & Sharing	Digital Integration (DI)	DI-01	Edge Integration and Interoperability
		DI-02	Security Services and Coordination
		DI-03	Data platform
		DI-04	Semantic Model Manager
		DI-05	Data Sources & Destination Manager
		DI-06	Data Consultation & Subscription Manager
		DI-07	ML Model Manager
		DI-08	MLOps Pipeline Manager
		DI-09	Application Deployment Service
		DI-10	Architecture Manager
		DI-11	Data Quality Services
		DI-12	Machine Learning at the Edge
Accurate Monitoring & Verification	Digital Twin (DT)	DT-01	Model Selection and Reduction
		DT-02	Energy Consumption Models (Energy Twin)
		DT-03	Process Models (Energy Twin)
		DT-04	Product Models (Digital Twin)
		DT-05	Online Lifecycle Analysis (LCA)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

		DT-06	Online Lifecycle Cost Analysis (LCCA)
		DT-07	Energy Performance Indicators
		DT-08	Modelling Data Robustness
Decision Support and Adaptation	Decision Support Systems (DSS)	DSS-01	Sustainable Production Planning
		DSS-02	Model Predictive Control (Renewable integration)
		DSS-03	Fault Detection and Diagnosis
		DSS-04	Context-Driven Visualisation of Performance Metrics
		DSS-05	Digital Readiness Assessment Tool & Indicator
Workforce Development	Digital Skills (DS)	DS-01	Standards-based Energy Auditing
		DS-02	Worker Profiler
		DS-03	Training match making
		DS-04	Community of Practice

Each functional group from the figure above is described with some more detail hereafter.

3.2.3.1 Data Platform

The data platform is where both stream and batch data are received, managed, stored, analysed, and served to the end-users or end-applications. More details on the design of this component are given under Section 3.3.

3.2.3.2 Data Management Services

This set of services are dedicated to the usage of the data platform. Within the configuration capabilities, we can find the data sources set up for both real-time data from sensors and historical data, the definition of semantic models (and custom ontologies) or the configuration of queries for subscription to a custom data source by an application.

3.2.3.3 Platform Management Services

The management services allow the platform managers to internally configure and deploy the data platform technologies, as well as deploying custom users' applications on top of the platform to have access to the data from the platform (for instance, a custom visualization tool for the energy consumption on the different manufacturing machines of a company). This block also includes the tools dedicated to the coordination and orchestration of all the platform elements as well as the security components.

3.2.3.4 Machine Learning Services

This set of services contains the tools dedicated to machine learning modelling and allow the management of the entire lifecycle of a model, from its initial design and selection to the full MLOps pipeline, including training, deployment, validation and monitoring steps, among others.

3.2.3.5 Performance Modelling

This block is where the digital twins for products, processes and energy consumption are implemented. In addition, lifecycle performance analysis capabilities are included. The elements in this block make use of data from the data platform as well as the machine learning services for modelling purposes.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

3.2.3.6 Decision Support Tools

Decision Support Tools are the set of elements that, given the data from the data platform and the outputs of the performance modelling tools, provide the user with tools that support enhanced decision making. In addition, the outputs of this block can be fed back into the performance modelling tools to tune the performance indicators and used as inputs to the adaption and control features.

3.2.3.7 Adaption & Control

This block contains functionalities dedicated to the retrofitting to the control elements of the production system to provide real-time adaption and control capabilities based on the analytics of the data from the data platform and the outputs from the decision support tools. In DENiM, for example, these control mechanisms will be applied to maximising the use of renewable energy generation. Possibilities to implement automatic control mechanisms in the pilots will be analysed in future steps of the project.

3.2.3.8 Planning Tools & User-centric Services

The tools collected under this block are the ones that do not directly make use of data gathered by the data platform. They are independent functional components that enrich the capabilities of the DENiM solution with additional manufacturing planning tools and human-centric services. In this block, there are five main components: the digital readiness assessment tool, the training match making and worker profiler, the energy auditing tool, and the community of practice.

As DENiM services are generally data-driven, emphasis is placed on defining a reference approach to the core data management concepts for energy efficient manufacturing systems. In the following sections of the deliverable, details of the relevant services (the orange blocks in Figure 13) are described. This set of services are the essential ones for the interaction and utilization of the data platform internal components by the users in a flexible, efficient and reliable manner. The technical details of the remaining components of the DENiM platform will be developed further as part of subsequent work packages and associated deliverables.

3.3 DENiM data management architecture

The data platform is the central element of the DENiM solution, as most of the functional blocks will either consume from or produce data to be stored in this platform (or both). In addition, the data platform will have external services dedicated to the interaction of the developers with the internal platform components. For instance, there will be services dedicated to the subscription of an application to a certain set of data sources or to the deployment of custom applications for data analytics on top of the data platform hardware.

This data platform is a critical element within the whole DENiM solution, and for that reason, this section is dedicated to the analysis of existing practices, requirements and architectural design of the data platform.

3.3.1 Current practices on data platforms

Many companies are aware of the need for a better understanding of their production processes. Instead of periodic reports, company managers are now requesting timely status reports for improvements to ensure the company's profitability. Traditionally, these analysis reports are created with static and inconsistent sources of information, being locked in spreadsheets or other applications (e.g., SCADA). Thanks to the advances in data architectures, it is possible to take closer control of their production processes and additional information about the company.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

The amount of data is growing exponentially per year, not only structured data but also unstructured data. To obtain value from this data, a set of tools is needed to transport, store and process it, and this is where a data architecture comes in, being the foundation of any data strategy. A data architecture is the process of standardising how organisations collect, store, transform, distribute and use the data (*What is Data Architecture? How to Drive Real Business Results through Data*).

Current practises in relation to data platforms can be classified into two main groups: those providing the data architecture with all the elements installed but without any configuration or those providing services using the data architecture. Within the first group, it is possible to highlight companies that offer this service, such as Cloudera, AWS, IBM, Microsoft Azure, Oracle, etc. These companies provide the necessary components to create a data architecture, but it is up to the company's data engineers to configure it to provide the required service. Most of the available solutions are cloud-based, and their services are not free, except for a limited number of them (e.g. Cloudera Hortonworks Data Platform (HDP)). Concerning the second group, fewer companies provide these services, as it depends on the desired use-case. One can highlight the data architecture provided by OSIsoft, which allows the collection, enhancement and delivery of time-series data in real-time from different devices (*OSIsoft | Operational Intelligence | PI System*). The use of this data platform is not free and the price is calculated based on the number of devices or services required. It also allows integration with SAP HANA, but again this integration is and additional expense.

3.3.2 Data architecture requirements

A data architecture is a key element in developing a digital twin-oriented platform. A data platform comprises models, policies, rules or standards that govern what data is collected and how it is stored, processed, integrated and used in data systems and organisations (*Data Architecture Characteristics and Principles - Huawei Enterprise Support Community*). Additionally, the data architecture must support the ingestion, processing, and storage of large amounts of data, known as Big Data. The characteristics of Big Data are commonly referred to as the four *Vs* (*What are the 4 Vs of Big Data?*):

- **Variety** refers to the many types of data sources and types of ingested data, which can be structured, unstructured, or some combinations of them.
- **Velocity** covers the speed with which data is generated, ingested, processed and used.
- **Volume** of data corresponds to the size of data that is going to be analysed, stored and processed.
- **Veracity** of data denotes the trustworthiness of the data. Data comes from a wide range of different data sources, and it is essential to understand the chain of custody, metadata and the context in which the data was collected.

The proposed DENiM data architecture is based on the Lamba-Architecture concept (described in the next section) must be able to ingest and process any type of data, both structured or unstructured (or some combination of both). The data is generated from multiple sources, so the architecture must be able to support the connectivity of industrial devices by using messaging protocols (e.g., MQTT, Modbus), integration with the current system (e.g., MES, ERP), ingestion from existing databases or external databases, and historical data uploaded from offline files. In addition, the architecture must be able to scale without code changes based on the amount of data ingested without the system degrading.

Data can potentially be ingested in the DENiM platform in one of two ways, depending on how it is generated:

- Online: The platform must be able to ingest, process and use real-time data in less than one second. This value has been selected as a compromise between hardware requirements and pilot needs (based on sensors acquisition frequency).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- Offline: offline data ingestion must also be supported (e.g., historical data or current databases).

In relation to the volume of data, the data architecture must be human and machine fault-tolerant, so it is necessary to have components that allow for distributed transportation, processing and storage. These components can be implemented in local, cloud, or hybrid cloud configurations. Concerning data veracity requirements, the data architecture must also support data quality and reliability checks to ensure the accuracy, consistency, precision, privacy, integrity and uniqueness of the ingested data in the architecture. Table 3 summarises the DENiM requirements that the proposed data architecture must satisfy, grouped by category.

Table 3: Main requirements by category must satisfy the proposed data architecture.

Data Architecture Main Requirements				
Variety	**Velocity**	**Volume**	**Veracity**	**Connectivity**
Supports any type of data	High throughput	Human and Machine Fault-Tolerant	Supports data quality and reliability tests (accuracy, consistency, precision, privacy, integrity, uniqueness)	Ingest data from devices through industrial communication protocols (e.g., MODBUS, MQTT, S7)
Manage raw data	Real-time ingestion	Distributed processing	-	Integration with current systems (e.g., MES, ERP, SCADA systems)
-	Low latency	Distributed storage	-	Ingest data from an existing data warehouse or external databases
-	Real-time processing (microseconds to seconds)	Multi-region deployment	-	Upload historical data from offline files
-	Batch ingestion	Scalable based on microservices	-	-
-	Batch processing (minutes to days)	Supports hybrid cloud	-	-

3.3.2.1 Mapping to the general DENiM architecture requirements

The data architecture requirements exposed above represent a step further in detail from the previous general requirements listed in D3.1. The mapping between those general requirements and the ones specific to the data architecture is as follows:

- **Variety** requirements can be mapped to the general requirement **DI-RQ-04**, which is dedicated to data interoperability. With variety, we refer to the ability to accept data in multiple formats and from disparate sources.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- **Velocity** requirements were captured in **DI-RQ-05**, which refers to adaptative stream processing in near-real-time.
- **DI-RQ-07** about scalable distribution and orchestration data processing can be mapped to the **volume** category, as it includes the hybrid-cloud support, scalability, distributed storage and processing.
- **Veracity** corresponds to **DI-RQ-06** about data integrity.
- Finally, **connectivity** maps back to **DI-RQ-01**, **DI-RQ-02** and **DI-RQ-03**.

3.3.3 Data architecture specification

A Lambda Architecture (LA) is a generic, scalable and fault-tolerant data processing architecture. One of the benefits of the LA, is its ability to satisfy the requirements for a robust system that is fault-tolerant, both against hardware failures and human errors. The proposed architecture comprises three layers (as shown in Figure 14) to serve a wide range of uses cases, aiming at simplicity, scalability, fault-tolerance, low-latency reads/updates (*Lambda Architecture » λ lambda-architecture.net*). A brief description of these layers is presented as follows and is described in more detail in the following sections:

- Speed layer: (or real-time layer) is responsible for performing ingestion, processing and transporting in real-time based on received data from different sources.
- Batch layer: is where raw data is stored and optionally processed if the application requires it. As the information is stored in the rawest format possible, it allows for batch processing. This acts as a store for an immutable master dataset, which is also available for other analytics services as required.
- Serving layer: interacts with end-users and services, exposing processed data in near-real-time from both the speed and batch layer.

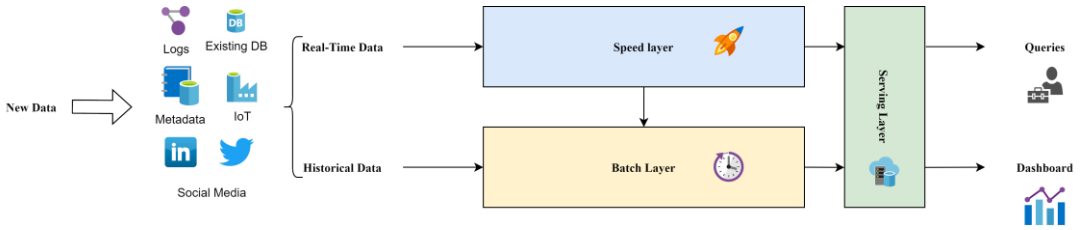

Figure 14: Generic approach of Lambda Architecture (LA).

3.3.3.1 Speed layer

This layer provides temporal storage of real-time data to prevent data losses in case of problems during transportation to other layers. Since the data acquisition from devices is estimated to be in microseconds/seconds, the processing is expected to be near real-time; such data processing will need to be fast and efficient, meaning that the platform must be designed to support concurrency. An outline of the structure of the speed layer is shown in Figure 15 and is described in terms of the required functionality below.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

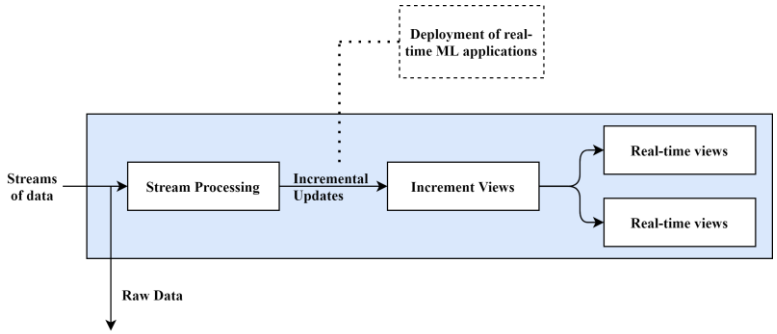

Figure 15: Speed layer components in Lambda Architecture.

The main goal of the speed layer is to produce views that can be queried in an effective way to transport data between layers. Increment views only represent the stream of data that was ingested/processed in the speed layer. Then, this data is appended into real-time views to be accessible for others layers. These real-time views optionally and temporally store (by the election of the end-user) real-time data.

3.3.3.1.1 Required functionality

The requirements of the speed layer are:

- It must support fast processing (event-processing and/or micro-batch processing) on ingested data streams.
- The speed layer shall transport and process the incoming data in low latency. The latency will be in microseconds in event-processing and seconds in micro-batch processing.
- Data ingestion and processing must be fault-tolerant, and this is based on a distributed approach.
- It shall support the ingestion of large amounts of data with no degradation problems.
- It must create real-time views to access the data quickly.
- This layer is responsible for transporting data to the other layers (batch-layer and serving-layer). It shall transfer stream data to the other layers.
- Regarding connectivity requirements, the speed layer must be able to ingest data from the following systems:
    - MQTT client. An MQTT client is any device that operates by using MQTT protocol and connects to an MQTT broker over a network (*MQTT Client and Broker and MQTT Server and Connection Establishment Explained - MQTT Essentials: Part 3*).
    - Programmable Logic Controller (PLC). The architecture must support the ingestion of stream data from PLC through a communication protocol (AB-ETH, ADS-AMS, BACnet/IP, CANopen, DeltaV, DF1, EtherNet/IP, Firmatea, KNXnet/IP, Modbus, OPC UA, S7).
    - If the real-time data does not come directly from the device but is first stored in a database, the data architecture must be able to ingest the data from the database in real-time.
    - Existing ERP systems.

These connectivity requirements correspond to the need to support legacy equipment integration as per requirement DI-RQ-01 and the integration of other data sources from DI-RQ-02 (new sensors, PLC controllers, or alternative business systems – ERP)

3.3.3.1.2 Functional description

This layer is in charge of ingesting, transporting and processing data in real-time. The functional architecture is shown in Figure 16 and described below:

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- Connectors to translate the communication protocol of devices to the language of the stream engine.
- Stream engine to transport real-time data.
- Stream processing engine (event-processing and/or micro-batch processing) to pre-process real-time data.
- UI interface to manage all the stream clusters.
- Service to maintain naming and data configuration to provide flexible and robust synchronisation across the distributed stream engine.
- Real-time ML service. ML applications that required real-time data (event-processing data) should be deployed in a micro-service that takes data from the stream engine. If the ML model needs data with a time resolution longer than 1 second, it will be deployed in an MLOps pipeline that takes data from the serving layer (as described in section 3.3.4.4).

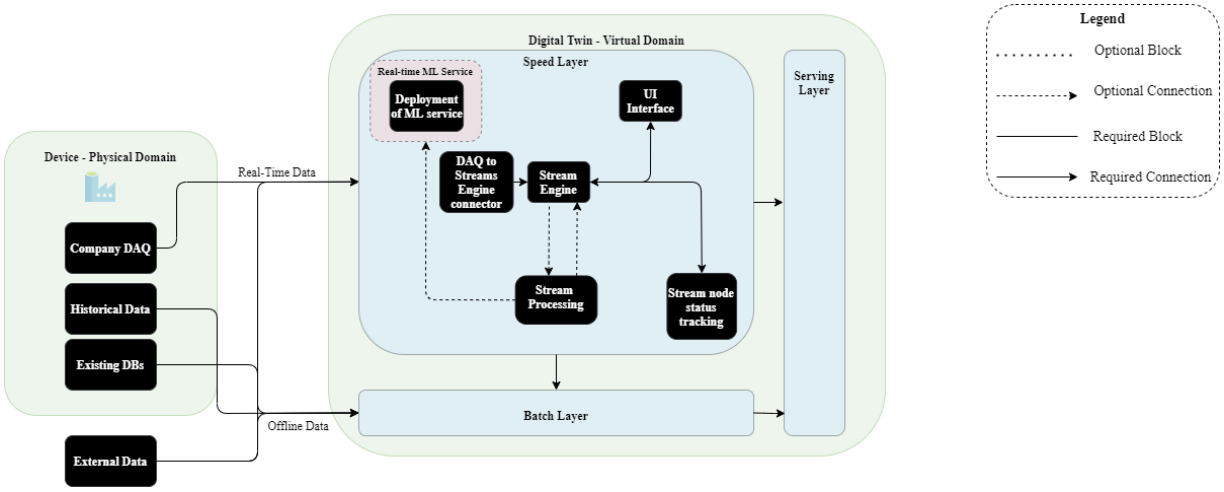

Figure 16: Functional description of speed layer in lambda architecture.

3.3.3.1.3 Available technologies

Different software tools can be used at the speed layer, with this layer being responsible for multiple tasks as discussed above. A comparison between the most popular ones can be found in Table 4.

Table 4: Comparison between tools for the speed layer.

Tool	Functionality	Open-Source	Processing	Programming Language	Distributed	Comment
Apache Kafka	Ingestion (Kafka connect)	Yes	–	Java, Scala	Yes	Majority of connectors not open-source; connectors limited; custom connectors required.
	Transporting messages (Kafka)	Yes	–	Java, Scala	Yes	
	Processing (Kafka	Yes	Event-Processing	Java, Scala, SQL	Yes	Stream processing

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

	Streams, KSQL/ksqlDB)					works within the cluster.
	Temporal Storage (Kafka)	Yes	–	Java, Scala	Yes	Customisable duration of storage.
Apache Zookeeper	Maintaining centralised configuration information	Yes	–	Java	Yes	Zookeeper is mainly used to track the status of nodes in the Kafka cluster and maintain a list of Kafka topics and messages.
RabbitMQ	Transporting messages	Yes	–	Java, Python, .NET, JavaScript, Rust, Scala, Erlang.	Yes	No support for message ordering and does not guarantee atomicity.
ZeroMQ	Transporting messages	Yes	–	C, C++, C#, Erlang, Go Haskell, Java, Node.js, Perl, Python, Ruby, Rust.	Yes	Not good for transaction-based systems. Easy to implement.
Apache ActiveMQ	Transporting messages	Yes	–	C, C++, C#, Erlang, Go, Haskell, Java, Node.js, Perl, Python, Ruby, Rust, Pyke, Racket.	Yes	Good performance if persistence is not a requirement. Supports transactions in messaging system.
Apache Flume	Collecting, aggregating, and moving data.	Yes	–	Java	Yes	Built-in support for various sources and sinks. Possible duplicate messages. Scaling the correct hardware size is a testing and failure process.
Apache NiFi	It supports powerful and scalable	Yes	–	Java	Yes	Dynamically configurable, customisable

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

	directed graphs for data routing, processing, transformation and system logic.					ingestion methodologies. Can define separate paths for the same datasets: a stream processing one and a batch processing one.
Apache Flink	Processing	Yes	Event-processing	Java, Scala	Yes	Low popularity. Support for exactly once processing. Accurate results for late coming and out of order data. Integrated ML library (Still with few approaches available).
Apache Storm	Processing	Yes	Event-processing	Clojure, Java	Yes	It supports any programming language.
Spark-Streaming	Processing	Yes	Micro-batch processing	Scala, Python, Java and R.	Yes	Near real-time processing, queries are processed using a micro-batch processing engine.
Amazon Kinesis	Collecting, processing and analysing	No	Event-processing	Java, Scala, Python	Yes	It's offered as a managed service in the AWS cloud, so it cannot run on-premise. It uses Apache Flink libraries.
Kafdrop	UI Interface	Yes	–	–	No	Web UI for viewing Kafka topics and browsing

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

						consumer groups.
CMAK	UI Interface and cluster manager	Yes	–	–	No	Tool for managing Apache Kafka cluster.
AKHQ	UI Interface and cluster manager.	Yes	–	–	No	Kafka GUI to manage topics, topics data, consumers group, schema registry, connectors.
Zoonavigator	UI Interface	Yes	–	–	No	UI interface to manage Zookeeper nodes.

There is a wide range of different tools to consider, and each has its advantages and disadvantages. The choice of one or the other will depend on the elements chosen in the other layers, as they must be compatible, and a connector is required to allow communication between these layers in a distributed and scalable way. This decision is not based on the selection of an individual element, it is about the set of tools that form a cluster for the stream layer, and it must support all of the requirements listed above. Apache Flink, Apache Flume and Apache Kafka could satisfy the proposed requirements, as they are powerful tools that enable ingesting, processing and transporting the streaming data. Within these technologies, we believe that a Kafka cluster would fit the requirements proposed above. The Kafka cluster is made up of various clusters of Kafka-brokers to transport the data, Kafka-connect cluster to link the brokers with other data sources and sinks, Zookeeper (and Zoonavigator as UI interface) to track the status of nodes and recovery in case of some nodes going down, Kafka-stream and/or KSQL to stream processing in real-time, and Kafdrop/AKHQ to manage and visualise the cluster. This decision does not mean that other options will not be implemented in the future, such as Spark Streaming for micro-batching real-time processing, but their choice will be determined by the elements chosen in the batch layer.

3.3.3.2 Batch layer

The batch layer is responsible for distributed ingesting, processing, and storing historical data from different sources: existing databases, historical files, speed layer, etc. For this layer, It is desirable that the data is stored in the rawest possible format in order to create a master data set based on an append-only strategy. The overall purpose of this requirement is to prevent the loss of data and to support the creation of new business opportunities in the future. The useful raw data is processed in batches in a distributed way and exposed to services as required.

3.3.3.2.1 Required functionality

The requirements of the batch layer are:

- It must support the processing of the stored raw data. However, this processing could take a long-time, so once it is processed, the batch layer shall transport the processed data to other layers to be exposed to other services at low latency.
- The processing, transporting and storing must be carried out in a distributed way to support fault tolerance.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- It has to be scalable without any code change, only adding new nodes to the platform and connecting with a coordinator node. These nodes could be deployed in local, cloud or hybrid cloud configurations
- Regarding connectivity requirements, the batch layer has to be able to ingest data from different data sources:
    - Historical data from existing devices.
    - Existing databases from current systems (e.g., MySQL).
    - Raw data from the speed layer.

3.3.3.2.2 Functional description

To ensure the correct functioning of these processes, this layer must contain the following functional elements (Figure 17):

- Connectors or software pieces that allow ingesting data from other data sources.
- Batch processing element to process raw data.
- File distributed system to store the data.
- A resource negotiator element used to manage resources among all the nodes and scheduling jobs.
- A processing engine element for batch processing over raw data stored in a file distributed system.
- A programming environment to implement processing that runs in the processing engine element. Different programming environments could be provided for data engineers to select the most suitable one for their respective programming skills and specific application.

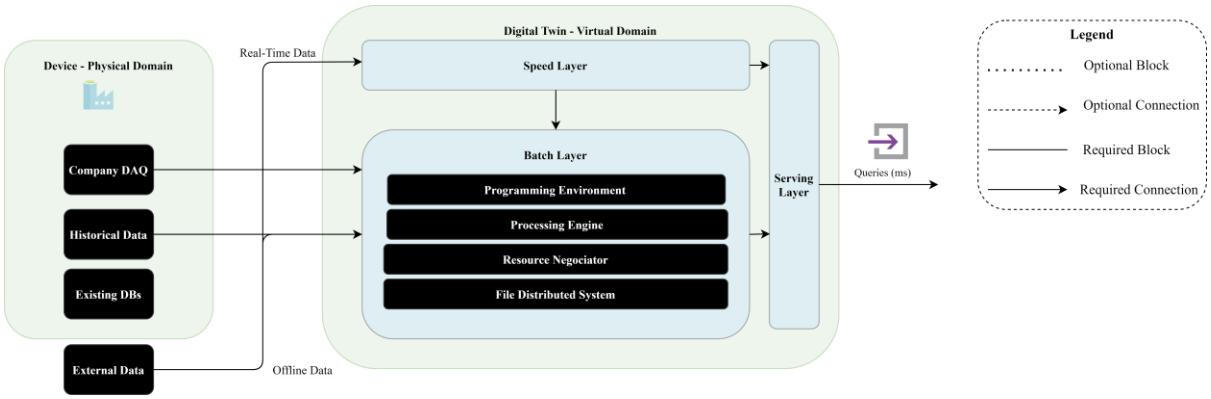

Figure 17: Batch layer in lambda architecture.

3.3.3.2.3 Available technologies

Table 5 shows a brief comparison of technologies that could be potentially used to implement the batch layer.

Table 5: Comparison between tools for batch layer.

Tool	Functionality	Open-Source	Processing	Programming language	Distributed	Comment
Apache Sqoop	Data acquisition	Yes	–	Java	Yes	On denormalised data, it runs map-reduce processes

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

						that can be time-consuming. Built-in connectors for different data stores.
Apache Spark	Processing	Yes	Batch-processing	Scala	Yes	It's easy to make use of Spark with API's in Java, Scala, Python, and R.
Apache Oozie	Pipeline creation and work scheduling	Yes	–	Java, JavaScript	Yes	Oozie is a workflow scheduler system to manage Apache Hadoop jobs.
Apache Zookeeper	Cluster management	Yes	–	Java	Yes	It is a coordination service for the distributed application that enables synchronisation across the cluster.
HDFS	Storage	Yes	–	Java	Yes	HDFS can store a large amount of information in a simple and scalable way.
Apache Cassandra	NoSQL Storage	Yes	–	Java	Yes	Key-value pairing. It uses CQL (Cassandra Query Language), which is an incomplete version of SQL. Not ideal for time-series data
Apache HBase	NoSQL Storage	Yes	–	Java	Yes	Good with random reads. Sits on top of HDFS.
Apache Hive	Data warehouse for query and analysis	Yes	Batch processing	Java	Yes	Translate MapReduce to SQL-like query language over HDFS Not suitable for real-time analysis

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Apache Pig	Analysing	Yes	Batch processing	Java	Yes	Pig jobs abstract MapReduce complexity. Provides extensibility allowing User Defined Functions that can be written in Groovy, Python, Java, JavaScript and Ruby.
Apache Tez	Processing	Yes	Batch processing	Java	Yes	It is based on MapReduce technology. It natively supports HDFS. More suitable for batch processing.
Apache Impala	SQL Query	Yes	–	Java, C++	Yes	Build-in support for HDFS and HBase.
Apache Zeppelin	Notebook interface for ingestion, discovery, analytics and visualisation	Yes	–	Java, Python, Scala, JavaScript	Yes	Allows any language/data-processing backend to be plugged into Zeppelin.
Amazon S3	Storage	No	–	Java presumably on the backend JavaScript, CSS on client-side	Yes	Mature product with ample support. Flexibility and security. Analytics on demand.
Google Cloud	Storage, processing, ML, management tools, security, analytics	No	Yes	Python, Java, Go, C++, Ruby	Yes	It is a collection of numerous cloud services that can be integrated with any personal or company-wide project.
Apache Mesos	Cluster management	Yes	–	Java, Python	Yes	Mesos can manage all the resources in your

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

						data centre but not application-specific scheduling.
Amazon EMR	Processing and analysing	No	Yes	Java, HiveQL, Pig, Cascading, Ruby, Perl, Python, R, PHP, C++, Node.js	Yes	EMR stands for Elastic MapReduce, and it is based on Hadoop.
Apache Airflow	Pipeline creation & work scheduling	Yes	–	Python	Yes	Using Python allows developers to import libraries and classes to help them create their workflows.

The most promising solution, given its maturity and compatibility with different elements, maybe the set of tools that comprise the Hadoop ecosystem, as it meets the requirements previously detailed. These elements are Hadoop Distributed File System (HDFS) as a distributed storage layer, Yet Another Resource Negotiator (YARN) for cluster resource usage planning and job management/scheduling, HIVE/Spark to translate SQL (or like-SQL) to Map-Reduce to process distributed data stored in HDFS, Zeppelin and/or Jupyter notebook as data-science/data-engineering platform to process and manage the data. Other elements could be further implemented in case of need, such as Zookeeper to maintain and recover the Hadoop nodes or other microservices to implement batch-processing over data stored in HDFS.

3.3.3.3 Serving layer

The serving layer is where the data in its final form is exposed to other services. Hence this is mostly processed data. It is responsible for merging data feeds from the speed and batch layers to perform low-latency queries. Exposed data could be exported via multiple protocols, depending on the chosen technology.

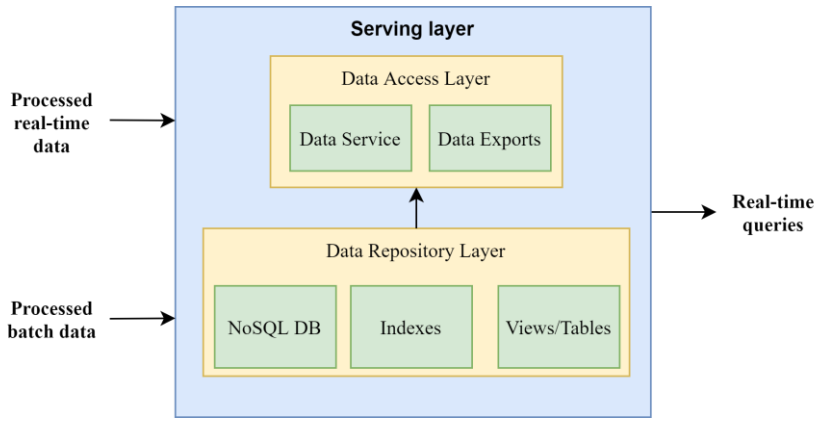

Figure 18: Serving layer in lambda architecture.

3.3.3.3.1 Required functionality

The requirements of this layer are:

- It must allow low latency queries over the exposed data.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- It has to be distributed and scalable without degradation problems to serve a wide range of use cases to various services.
- It shall be fault-tolerant, allowing the functionality not to be stopped in case some nodes go down.
- Data must be exposed by fast access/high performance in near real-time to other services.
- It must support serving multiples data models (same data with different pre-processing for different use cases).
- It is required to be deployable on-premise, cloud and hybrid cloud.
- It must support the ingestion of data from batch and speed layer in near real-time.

3.3.3.3.2 Functional description

The functional description of this layer is very diverse and depends on the chosen technology. It is composed of three different functional blocks (see Figure 18):

- Query module - handles queries from other services.
- Ingestion module - responsible for managing data ingestion from batch and speed layers.
- Storage module - stores all the queryable data. There are different approaches concerning this module. The storage module of the batch layer may be re-used to keep the serving layer's data, or the serving layer may contain its own storage module. In general, the first option will be used to avoid hardware resources redundancy and take advantage of the robustness and distributed nature of the batch storage (HDFS).

3.3.3.3.3 Available technologies

Different databases could be chosen for this layer, and it depends on the requirements previously presented.

Table 6 contains the advantages and disadvantages of the most commonly used databases.

Table 6: Comparison between software tools for the serving layer.

Database	Advantages	Disadvantages	Comments
Apache Cassandra	Native Spark-Cassandra connector	CQL (not SQL a query language)	Key-value database
	Queries are performed in memory	No allows to ACID transactions	NoSQL database.
	Compatible with Presto		Not natively designed for time series, although it can be adopted as a time-series database.
	Distributed		Slow responses in comparison with other time-series databases.
	Easy scalability (multi datacentre deployments)		
	Replication across other nodes		
InfluxDB	Initially designed for time series	The most of connectors with other software elements are paid.	It is the best database for time series, but compatibility with the most used tools for speed/batch layer is not entirely trivial.
	SQL-Like query language	Full functionality is not open-source	The basic use is open-source, but the completed functionality is only in the paid version.
		Not natively compatible with another external query engine like Presto. Although, it is	Easy to get started. However, it is pretty rigid and limited, with no ability to create additional indexes,

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

		possible creating a custom connector.	update metadata, enforce data validation.
		Not support unstructured data from other sources.	It is no possible to ingest data that is not time-based.
TimescaleDB	Short-learning curve	Although it is possible to integrate it with our data architecture, this connection is not given by default, and it may not be as trivial as it seems.	Based on PostgreSQL, and offers the best of NoSQL and relational words for time-series data.
	Based on Postgres. Any tool or extension that works with PostgreSQL works with TimeScaleDB.	It is not possible to ingest data that is not time-based.	Each time-series measurement is recorded in its own row, with a time field followed by any number of other fields, which can be floats, ints, strings, Booleans, arrays, JSON, geospatial dimensions, time-stamps, etc.
	Other external engines support it by using PostgreSQL connector.		For workloads with very low cardinality (e.g., 100 devices), InfluxDB outperforms TimescaleDB.
			For workloads with moderate to high cardinality (e.g., 100 devices sending ten metrics), TimescaleDB outperforms InfluxDB.
Apache Druid	Druid is designed for workflows where fast ad-hoc analytics, instant data visibility, or supporting high concurrency is essential.	It is not as widely used by the community as other databases such as Cassandra, MongoDB, etc.	Open-source analytics data store designed for sub-second OLAP queries on high dimensionality and high cardinality data.
	Druid streams data from message buses such as Kafka and AmazonKinesis, and batch load files from data lakes such as HDFS and AmazonS3.		Easy integration with Kafka-Hadoop ecosystem.
	Native connector to external queries engine (e.g., Presto).		The community makes some open-source tools for Apache Druid. E.g., Turnilo for data visualisation in real-time.
	Combining streams and historical analytics (Batch and Real-Time ingestion).		Historical data could be stored into HDD disk on servers, streaming data could be stored into SSD disk to fast queries.
	SQL is used to query data		
KairosDB	It could be deployed in public, private, and hybrid clouds		
	Fast time series databased build on top of Cassandra	Poor community support in GitHub-public projects.	
	Share the same Consistency-Availability properties with Cassandra	Not native connector with other external queries engine (e.g., Presto)	
Prometheus	Powerful easy to use monitoring	Prometheus does not provide a dashboard solution – Prometheus' interface is	

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

		intended for ad-hoc debugging.	
	Flexible query language	Hadoop-Prometheus connector is not natively designed.	
	Active and responsive community	Not open-source connectors with other ingestion tools.	
MongoDB	Flexible database	High memory usage	Not oriented to time-series data
	High availability	Joins not supported	
	Easy environment setup	Limited data size	
	Scalability	No transactions	
	Flexible schema		

The selection of the appropriate database, in this case, is strongly dependent on the component selection in the other layers, as it is a requirement that they are fully compatible. Additionally, although it is not a requirement, it is recommended that this database be compatible with an external query engine, such as PrestoDB or TrinoDB. Given the requirements proposed in the previous point, InfluxDB, TimescaleDB and Apache Druid are potential fits for the proposed approach. Apache Druid is well suited for high volume, interactive and real-time analytics at a large scale, supporting distributing storage and deployment in public, private and hybrid clouds. The connectivity requirements from other layers with Apache Druid are natively supported and other external queries engines. It is not only designed to deal with time-series data. In addition, it is created and maintained by Apache, so there is a guarantee that it will remain open-source, which is not the case with InfluxDB.

3.3.4 Platform services

The core data platform will be enhanced with a set of platform services that will enable user interaction with the platform. The services consist of a series of software components deployed together with the data platform internal elements, each containing the necessary functionality to configure or obtain data from the platform. The functional representation of this set of services together with the data platform itself are presented in Figure 19, which is an extraction from the general architecture design presented in Section 3.2.

The primary set of required services is briefly described. However, more services may be identified and added in later development phases of the DENiM platform.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

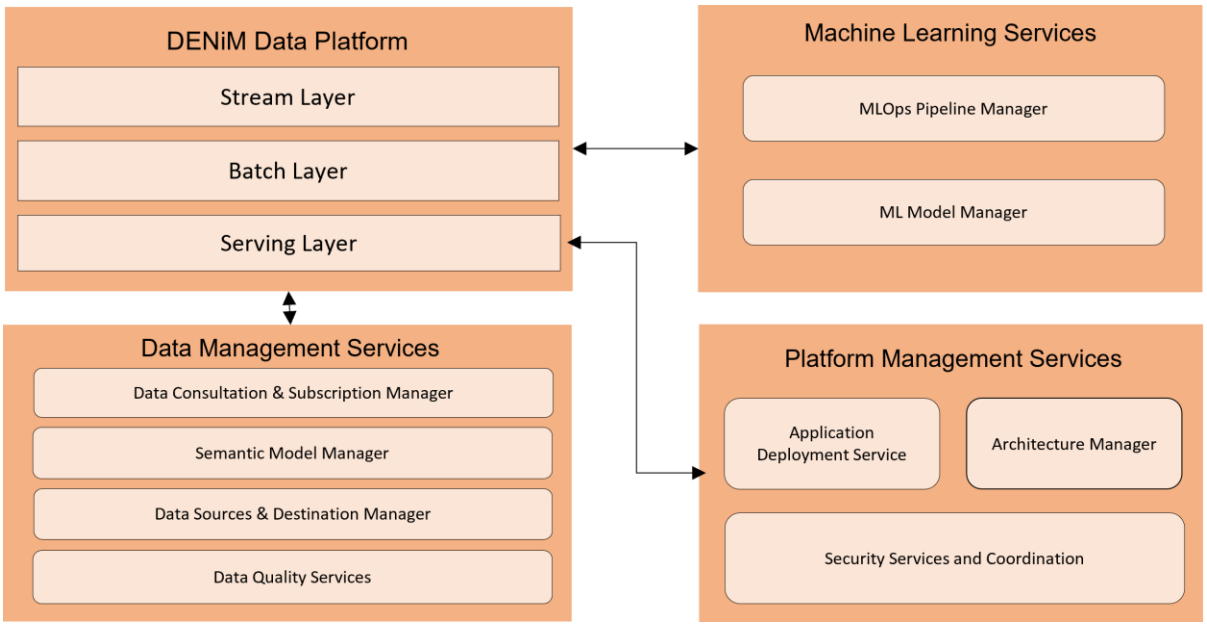

Figure 19. Data platform and platform services schematic

3.3.4.1 Semantic model manager

The Semantic model manager will consist of a tool that allows defining the semantic model of the use case. This way, the data sources can be mapped to the semantics and internally, all the data is semantically characterised, ensuring seamless data exchange and interoperability capabilities. It will provide the following functions:

- Using predefined ontologies based on industry standards but enhanced for DENiM's specific needs (most likely, SAREF-based ontologies).
- Definition of a custom ontology.
- Ontology instantiation to describe use case instances.

The deployed instances of the ontology are then stored on a dedicated database and used across the whole data management cycle to characterise the acquired and processed data fully.

3.3.4.2 Data sources/ destination manager

The objective of this service is to allow the user to configure the data sources for the data platform, which can be either real-time data (i.e., from sensors) or historical data from external databases. The user can connect data with several communication protocols (e.g., MQTT or Modbus TCP) or data from raw files (.xls, .csv) or existing databases. The tool also supports the possibility to define data sinks for the outputs of some services by using the same type of communication protocols.

The tool's output is the configuration of the data platform internal elements (connectors) to ensure that the configured data is correctly acquired.

3.3.4.3 Data consultation & subscription

This tool shall allow the user to configure the acquisition and subscription to the platform data with a query-like interface. This will be useful for implementing applications that require constant data streams, like a Digital Twin. Internally, this service interacts with the platform serving layer, where both real-time and batch data are available, and both raw and processed data can be acquired.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

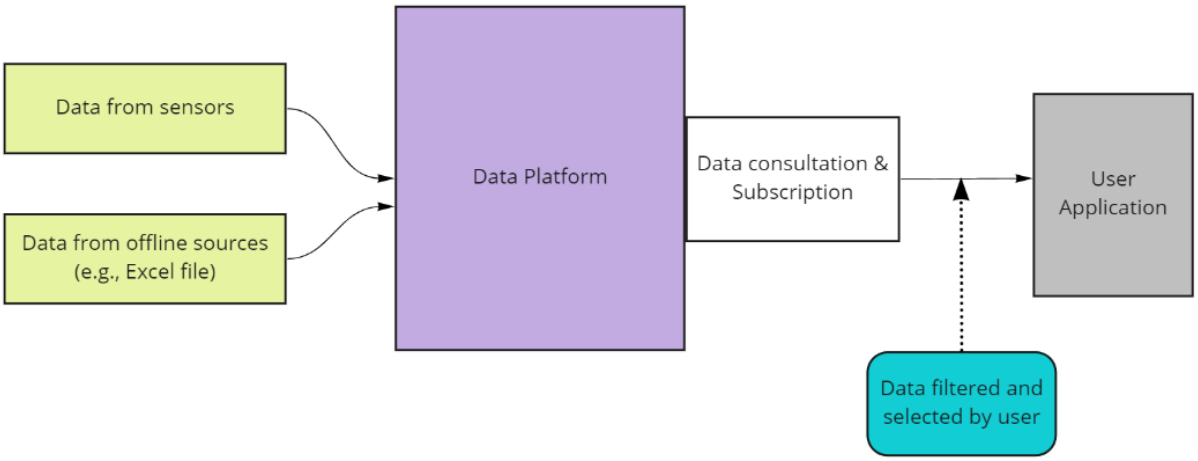

Figure 20: Schematic of the Data Consultation & Subscription manager integration with the platform.

Figure 20 shows a simple schematic representation of the integration of this service within the platform as a whole. The service is deployed at the output side of the data platform, configuring what data is presented to the end-user application (or to a single user performing a query). Any service or application requiring data from the platform will use this service as an intermediate element to configure the data acquisition according to their needs.

3.3.4.4 MLOps pipeline

The MLOps pipeline is the proposed solution to the needs identified in requirements DT-RQ-01, DT-RQ-02 and DT-RQ-03. First, DT-RQ-01 recognises the need for modelling approaches for data computation and predictions that are enriched with real-time data. Second, DT-RQ-02 requires methodologies that reduce the complexity of models selection, development, validation and deployment cycle. Then, DT-RQ-03 sets restrictions on the need to ensure that the quality of models is maintained throughout their lifetime. MLOps gives solutions to these three requirements, as exposed next.

The objective is to provide a platform that manages the infrastructure and processes for successful Machine-Learning (ML) development at scale. MLOps aim to unify the development and operation of ML systems, building a pipeline of micro-services to automate and monitor all the steps of an ML system (Google, n.d.). Figure 21 shows the elements that comprise an implementation of an MLOps pipeline.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

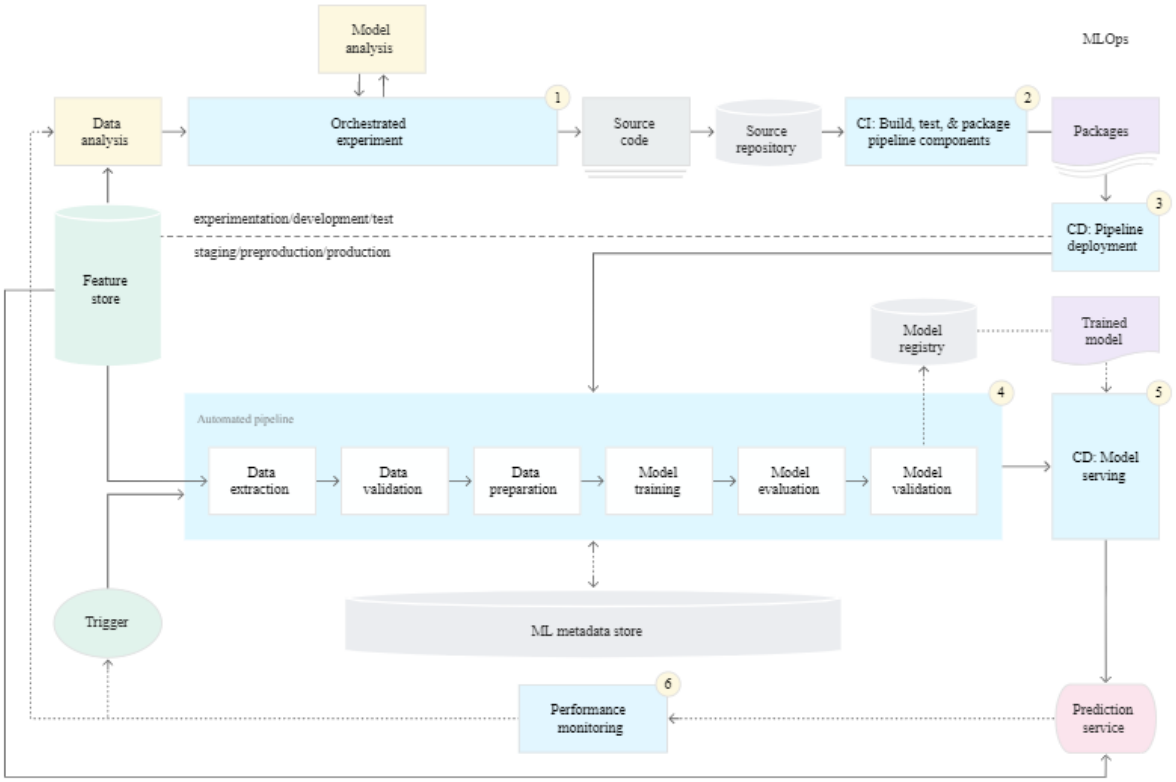

Figure 21: MLOps workflow process in an ML project (*MLOps: Continuous delivery and automation pipelines in machine learning*, no date).

The previous diagram numbers the stages of an MLOps pipeline (*MLOps: Continuous delivery and automation pipelines in machine learning*, no date):

1. Development and experimentation of different Machine-Learning approaches to the given problem. The output of this point should be the source code of the ML approach.
2. Establish continuous integration (CI) building a source code, and implementing various tests. The outputs of this stage consist of packages, executables and artifacts that will be further deployed in other stages.
3. Set continuous delivery (CD) to deploy the artifacts previously established in the CI stage. The output will be the deployed pipeline with the new ML approach.
4. This deployed pipeline starts based on a trigger signal or a schedule. This stage produces as output the trained model that is pushed into the model registry.
5. After the automated pipeline starts, the trained model is ready to serve queries. The output of this stage is the deployed ML model.
6. This stage is dedicated to collecting metrics on the model performance working in real conditions. The output of this module is a trigger signal which generates warnings or is responsible for starting a new cycle in the pipeline.

3.3.4.5 ML model manager

The objective of this service is to provide a tool to define the requirements and functionalities of the MLOps pipeline manager. It can be understood as being a decoupled service supporting the MLOps pipeline manager or phase 0 of that service. The main target of this service is to ensure developers use a detailed methodology for the modelling strategy definition phase. The outputs from this planning process will be introduced in the service as plain text for documentation purposes. It covers the following aspects:

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

- Definition of the model inputs and outputs linked to data sources from the Data Source Manager. The MLOps pipeline may filter down the inputs to be used with feature engineering techniques.
- Definition of the problem to solve, establishing the minimum accuracy level below which the ML-model becomes impractical.
- Establish a baseline model based on the literature review. It is helpful to compare the results of the deployed model in the MLOps pipeline.
- Define the metrics to evaluate the model performance and the thresholds to detect the model decay. This threshold serves as a trigger to start re-training the models.
- Set up a strategy to put the trained model into production (e.g., shadow mode, canary deployment, blue-green deployment).
- Identify the monitoring metrics to be used for the model and server during the deployment stage.
- For large datasets, we must also define the sampling strategy that is used to extract data from the data platform (e.g., probability sampling methods or non-probability sampling methods).
- Determine a strategy to deal with concept drift and data drift.

It is important to note that these definitions and strategies can be adapted over time and should be set iteratively.

3.3.4.6 Application deployment service

This service allows the end-user to deploy custom applications that interact with the platform's data and the platform itself. This service will provide a template Docker File where the user can implement the configuration and deployment of the application. After that, these files can be uploaded together with the Docker File so that the platform can both hold and run the software application.

An example of an application to be deployed could be a user-defined dashboard for the data acquired and processed by the platform (e. g., the energy consumption of the factory floor machines).

The functionality of this service is mainly based on Docker, a set of PaaS products that allow serving software in the form of packages, called containers. Docker uses OS-level virtualisation, isolating the behaviour of the software from the hardware where it runs by providing standalone containers created from OS images. Figure 22 contains a schematic with the simplified process for creating a Docker Image and deploying it using a container. The Docker File is where the user specifies the source image and the operations to be run on top of it to create the custom Docker image. Those operations usually consist of copying the application source code into the container, setting up the network interfaces, and the required storage volumes, etc. Once the image is completed, the application starts running inside the container.

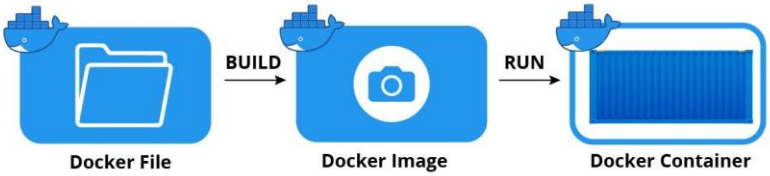

Figure 22: Basic process of creation of a Docker Image and deployment into a Container.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

By using Docker, our application deployment service can allow the user software code to work seamlessly in the data platform hardware, no matter what OS or tools the user employs for the development. In addition, Docker allows the setting up of custom networks for the containers' interconnection, which is crucial in order to extract data from the platform.

3.3.4.7 Architecture Manager

This service allows the platform manager to register and manage endpoints (deployment resources) and deploy the DENiM platform elements, both internal elements dedicated to data management and the DENiM tools and services to be developed as part of the overall DENiM platform. In addition, the deployment stacks (set of containers deployed as a single entity) can be dynamically deployed on top of the registered endpoints. An endpoint can be a simple Docker standalone instance, a Kubernetes cluster, a Docker Swarm instance or, in more advanced versions of the tool, custom cloud services like an AWS instance. Alternatively, deployment stacks can be a single container or a set of them that must be deployed together to work.

Figure 23 shows an example set of stacks and endpoints being deployed on several endpoints. Each stack can be made up of several Docker containers (for instance, the Data platform Speed Layer has containers for the Kafka brokers, the Zookeeper coordinator, the connectors for external sources, such as MQTT or PLC, etc.). At the same time, each endpoint can be a single Docker Standalone instance or a complex Kubernetes Cluster and can allow for the deployment of single or multiple stacks, depending on hardware limitations or networking requirements. The endpoints statistics can be consulted to decide where to deploy new stacks or if a change is needed in the configuration.

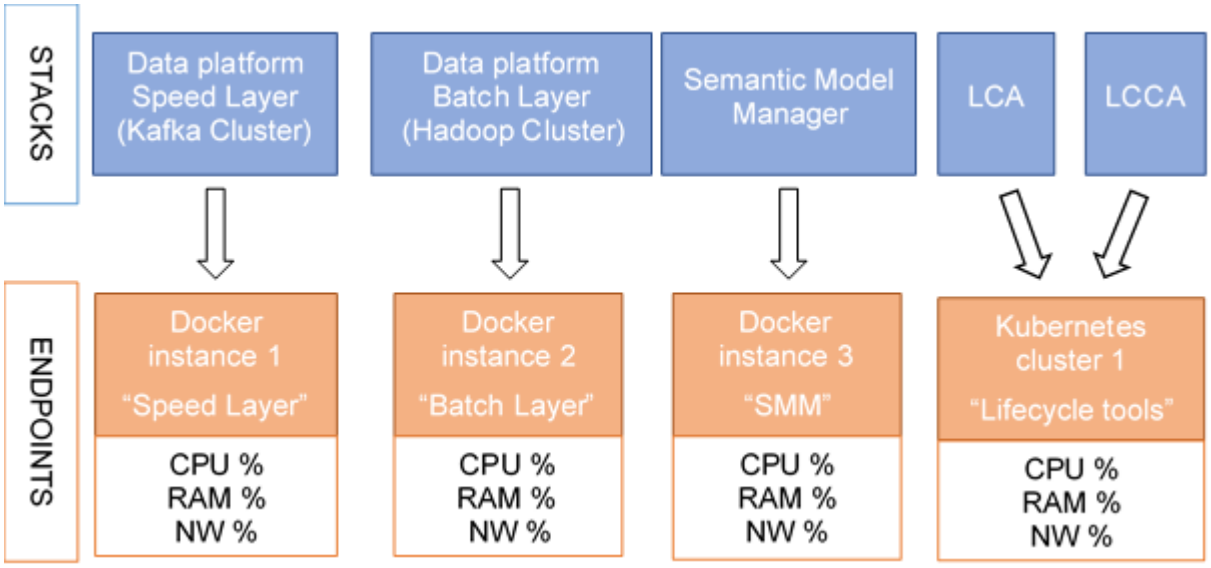

Figure 23: Example stacks and endpoints of the architecture manager.

3.3.4.8 Data security manager

Data security refers to the act of protecting data from unauthorised access and data corruption throughout its lifecycle. Data security covers various aspects such as data encryption, hashing, data governance, authentication, authorisation, etc.

This service will provide the required tools to manage data security across the whole platform and services. The specific mechanisms to be implemented will be defined in task 4.2 and reported under deliverable D4.2 Guidelines and Tools for Secure Data Integration.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

3.3.4.9 Secure edge integration and interoperability manager

This service will be used on top of the data platform for a secure and seamless connection with edge devices. Its implementation will be based on the Dazle platform (described further in 3.3.4.9.1), a commercial solution provided by the DENiM partner BAG-ERA. This middleware already contains several advanced mechanisms to manage both the integration and the security aspects, and the DENIM project is an opportunity to add specific modules in the frame of the use case pilots.

The DENiM architecture has to integrate numerous external hardware and software components. These components are mainly heterogeneous, mixing different technologies and protocols. This heterogeneity may also come from the fact that they belong to different technology generations, which is common in the industry domain. Finally, legacy components, ad-hoc configurations of some components or proprietary and/or closed formats prevent the usage of classical connectors. This heterogeneity can render the integration quite complicated.

The second point concerns security, which may not be equal for all components. The reasons are multiple: bad design, insufficient onboard capacity (e.g., some low-power microcontroller-based devices), older components that have little to no security protection.

The role of the secure edge integration and interoperability manager is to encapsulate all these components in such a way that they all behave in an appropriate manner for the other layers of the global architecture. Thus, even if the data sources may have some weak security guarantees, the encapsulation protects the upper layers of the platform to have to manage it. The secure edge integration and interoperability manager acts as a single trusted data source that provides information about the different devices and/or data software components to the upper layer.

Indeed, even if other layers can provide connectors to different standards, not all the connectors are available, and as they come from different actors, they are not all uniform in terms of security.

Thus, the presence of an integrated connectivity and security manager layer simplifies the global architecture and provides a clear separation of concerns.

The secure edge integration and interoperability manager layer has to manage:

- **Isolation**: each data source will be encapsulated and isolated from the others.
- **Uniformity**: each data source will be complemented if required in order to provide the same level of trust to the next layer
- **Input and Data validation** (detection or suspicion of ill-formed inputs): each data source will be analysed in real-time to analyze if the source is reliable (or not), if the data has not been corrupted and if no data has been lost. It is a detection mechanism when the fault is clearly identified and suspicion when the constraint of the underlying system (mainly asynchronous) makes it impossible to be sure of the fault. For instance, data loss when the frequency of emission is not known, and no additional mechanisms such as numbering, history or blockchain are used.
  - The corruption is related to the fact that the data, or in general, the behaviour of the component is not correct. For instance, a bad value could be sent to trigger further damages in the system. Or extra messages could be sent to saturate the system in DoS (Denial of Service) attack or to create an overuse of the battery in order to reduce the lifetime of a sensor. The cause can also be due to some internal malfunction such as:
    - calibration problem where the value drifts out the normal operating range,
    - battery problem preventing the correct reading of the sensed value,
    - interference that makes the frame unreadable for the receiver gateway,
- **Authentication**: a data source needs to have its identity proven by some means (another trusted entity or some credentials). In general, it prevents a fake component from accessing the system and submitting false information, modifying information, or getting access to some information.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

For DENiM, only the first case is considered. The other two operations are not allowed by design at this level.

- **Secure communication**: The communication is secured using asymmetric encryption and integrity checks, but also the end-points are authenticated through certificates. This enforces an end-to-end secure channel in between the data source and the connector of the platform.

The previous points are taken into account and take advantage of the mechanisms included in Dazle, which offers in addition to point encryption, strong authentication and repudiation capabilities that allow rejecting components suspected to be corrupted.

3.3.4.9.1 Proposed architecture

Dazle is a lightweight rule-based middleware that allows for a decoupling in time and space the components of a system of systems. In the DENiM case, Dazle acts as a mediator between physical systems and data sources, and the *speed and batch layers* described above.

Dazle utilises what is termed "bag" to encapsulate an arbitrary component in a unified way. A bag contains resources which in the case of a data source corresponds to the data emitted by this data source. The system may interact with the component indirectly when a resource is added to the bag. This mechanism provides the decoupling in time and space and simplifies the integration of legacy, third-party components, and in general, components that are not natively designed to work together. For instance, if the bag encapsulates a database, by adding or removing a resource, you may update the records of this database. If the component is an actuator, you may trigger some physical actions, such as shutting down a machine. The bag can encapsulate any type of component (both hardware and software).

Resources are accessible through a very simple interface based on three basic operations read, get and put

- Read allows retrieving from the bag resources corresponding to a given pattern
- Get allows verifying the presence of a given resource and consume it in an atomic way
- Put allows adding a new resource within a bag. More details are in (Pacull, 2017) and (Louvel and Pacull, 2014)

These primitives are used through a rule-based language that allows a usage scenario to be described by combining the discovery, the consumption and the creation of resources in a collection of bags. This rule-based language and the Dazle runtime enforces strong semantic representation based on the theory of Petri-nets (Sylla, Louvel and Rutten, 2016). As the run-time model includes transactional properties, any scenario proved as correct according to the Petri-net theory is guaranteed to be correctly executed by the Dazle rule engine. This enables the collection of data in a secure manner from the data source, and to make *on the fly* verification checks concerning the quality of the data, including some correlation with the current state, and to sink the refined verified data into the speed and batch layers. The rules can be dynamically added, removed, enabled, disabled or applied under specific conditions. This enables the adaption of the behaviour of this layer based on the current context.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

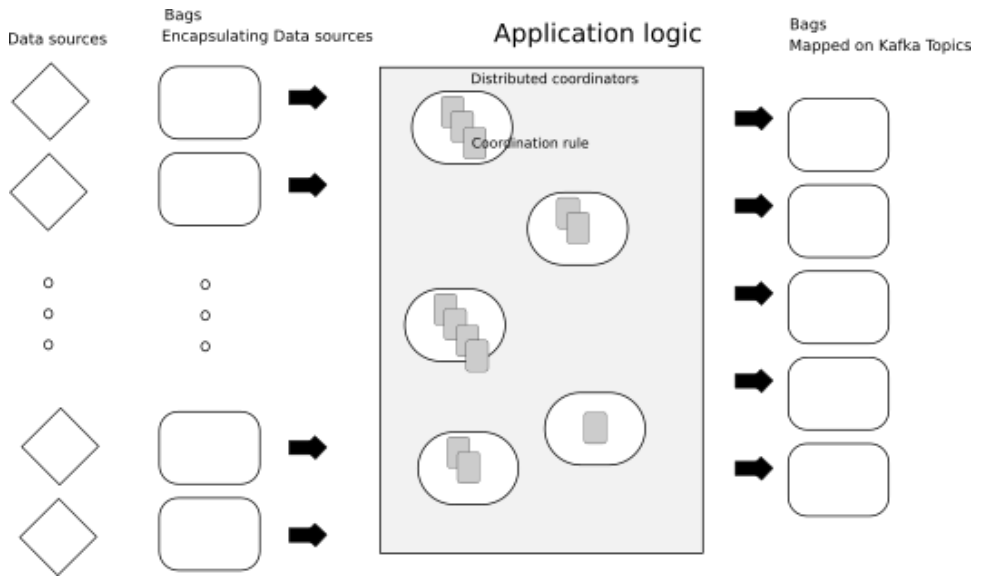

Figure 24: Secure edge integration and interoperability manager architecture.

Figure 24 shows the internal architecture of the secure edge integration and interoperability manager. On the left-hand side, we see an arbitrary number of data sources that are encapsulated within bags that provide the necessary adaptation to ensure compliance with the associated data source protocols and the security requirements. On the right-hand side, we find the bags mapped to the different Kafka Topics. A bag can be mapped to a single or several topics or to have several bags mapped to the same topic to adapt to the best configuration in terms of isolation, performance, redundancy, etc.

In the middle, a set of distributed coordinators enact coordination rules responsible for connecting the left bags with right bags with possible verification and transformation of the data. The internal connection is encrypted end-to-end. Rules and coordinators can be dynamically adapted to comply with redundancy, scalability and load balancing.

It is important to remark that this module allows pre-processing (e.g., data quality analysis), which could also be implemented directly at the batch layer. In the same way, the defined architecture for the data platform also includes securitisation and heterogeneous data source access.

3.4 Industry 4.0 reference architectures and mapping to the DENiM platform

Currently, there are two main reference architectures that act as standardisation approaches for Industry 4.0: the Reference Architecture Model for Industry 4.0 (RAMI 4.0) and the Industrial Internet Reference Architecture (IIRA).

The RAMI 4.0 architecture can be seen as a three-dimensional space whose axes represent (i) the value chain and lifecycle, (ii) the hierarchies of a production system and (iii) the layers that describe the physical asset, the integration of software and hardware components, communication capabilities, information creation through data, functional properties and business processes.

The IIRA architecture model presents five functional domains mapped against system characteristics and cross-cutting functions. These two reference architectures are highly correlated with each other, being complementary more than conflicting. The IIRA focuses on applicability and interoperability, while RAMI 4.0 is more focused on the digitalization of Smart Manufacturing.

Both reference architectures were designed to support standardised methodologies and help developers in the IIoT domain understand the most important concerns related to architectural requirements and create interoperable products across the industry. For this reason, the DENiM

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

platform will take these two architectures as a reference for the initial design phase of the DENiM architecture.

The DENiM platform is underpinned by the core data architecture with DENiM tools, services and applications being built on top of it. The result is a flexible and functional IIoT architecture with the main purpose being providing a digital thread to support the development of digital twin driven models and tools focused on energy and environmental impact assessment. The mapping of the DENiM architecture with the RAMI 4.0 and IIRA reference models is based on the following considerations:

- Real-world sensors, IoT devices, machines, etc., that the platform interacts with correspond to the **asset layer** in the RAMI 4.0.
- The **communication and integration layers** of RAMI 4.0 provide communication standards for data, services and control commands that link the physical domain with the digital capability. In the IIRA, the connectivity function refers to standards such as OneM2M and DDS. In DENiM, this maps to the **secure edge integration and interoperability manager** and the **semantic model manager**.
- The **information layer** in RAMI 4.0 refers to the services and data used, generated or modified through the asset. The IIRA has the corresponding management function dedicated to data management. In this layer, we can find the **data consultation and subscription manager**.
- The **functional layer** in RAMI 4.0 is dedicated to logical functions and corresponds to the analytics function in IIRA. In the DENiM platform, these functional and analytics capabilities are mainly addressed by the **model manager** and **MLOps pipeline**, together with the internal data platform services dedicated to data analytics and processing.
- The **business layer** in RAMI 4.0 enables the creation of business-level functionalities and business models under specific constraints. In IIRA, the intelligent and resilient control function responds to a similar objective. The DENiM platform will address this need through the **architecture manager** (which allows for the control and orchestration of the whole platform) and **decision support tools.** With a view to extensibility an **application deployment service** is used to enable DENiM developers to deploy their own services on top of the core DENiM data architecture and perform business-level analytics and operations, enhancing the core data services with both general and use case-specific applications. Furthermore, all the additional functionalities external to the data platform can be considered part of this business layer.

As demonstrated above, the DENiM platform architecture has a close relationship with both IIRA and RAMI 4.0 reference models.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

4 Conclusion

This deliverable provides an initial specification of both the semantic modelling and the data architecture for the DENiM project. These two aspects are highly interdependent, and their requirements and specifications are fundamental pillars for developing the core IT infrastructure of the DENiM digital intelligence solution. The DENiM functional blocks will be designed and deployed utilising this common data infrastructure, leveraging its services and either consuming or providing data to the central data platform.

In section 2, the semantic modelling strategy was defined based on an analysis of the semantic requirements and the study of the existing practices in the industry. A similar approach was utilised for the architecture specification discussed in section 3. These proposed services are inherent to the data platform itself, meaning they are the minimum set of services that allow platform managers and users to interact with the architecture in a configurable, extensible and adaptable manner. Some of the functionalities provided by those services are: deploying applications, configuring the platform semantics and data sources, consuming data or providing and storing it or designing and executing machine learning models with data from the platform.

The design of both the semantics and the architecture has been done with the intention of making it usable across a diverse set of industries. For this reason, a design decision was made to keep semantic models and ontologies as open as possible, letting the users define their own semantic models for each use case. However, as this can be a complex task, several ontologies will be included as templates to form the basis for development. An extensive analysis of the project needs was done to select these templates through the analysis of the DENiM tools and services in terms of data interaction requirements. With these outputs, a list of fields for semantics and a set of requirements for the architecture design were generated.

The outcomes of this deliverable, together with the pilot focused requirements analysis conducted as part of deliverable D3.1, will form a basis for the evolution from requirements and specification to components design and development in subsequent work packages.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

References

AI@Edge Community (2021) *What is ONNX?* Available at: https://microsoft.github.io/ai-at-edge/docs/onnx/ (Accessed: 18 May 2021).

Alexopoulos, K. *et al.* (2021) *Predictive maintenance technologies for production systems. A roadmap to development and implementation*. Available at: http://foresee-cluster.eu/wp-content/uploads/2021/07/ForeSee-roadmap-to-the-predictive-maintenance-technologies-for-production-systems-v1.0-final.pdf (Accessed: 16 July 2021).

Apache Software Foundation (2021) *Apache Jena*. Available at: https://jena.apache.org/ (Accessed: 20 May 2021).

Barbau, R. *et al.* (2012) 'OntoSTEP: Enriching product model data using ontologies', *CAD Computer Aided Design*, 44(6), pp. 575–590. doi: 10.1016/j.cad.2012.01.008.

Borsato, M. (2017) 'An energy efficiency focused semantic information model for manufactured assemblies', *Journal of Cleaner Production*, 140, pp. 1626–1643. doi: 10.1016/j.jclepro.2016.09.185.

Data Architecture Characteristics and Principles - Huawei Enterprise Support Community (no date). Available at: https://forum.huawei.com/enterprise/en/data-architecture-characteristics-and-principles/thread/713297-893 (Accessed: 18 May 2021).

Eclipse Foundation (2021a) *Eclipse Papyrus*. Available at: https://www.eclipse.org/papyrus/ (Accessed: 20 May 2021).

Eclipse Foundation (2021b) *Eclipse RDF4J*. Available at: https://rdf4j.org/ (Accessed: 26 May 2021).

Elasticsearch B.V. (2021) *Elasticsearch - The heart of the free and open Elastic Stack*. Available at: https://www.elastic.co/elasticsearch/ (Accessed: 20 May 2021).

ETSI (2019) *SAREF4INMA: an extension of SAREF for the industry and manufacturing domain*. Available at: https://saref.etsi.org/saref4inma/v1.1.2/ (Accessed: 10 June 2021).

ETSI (2020) *SmartM2M;Smart Applications; Reference Ontology and oneM2M Mapping*. Available at: https://www.etsi.org/deliver/etsi_ts/103200_103299/103264/03.01.01_60/ts_103264v030101p.pdf .

ETSI (2021a) *SAREF: the Smart Applications REFerence ontology*. Available at: https://saref.etsi.org/core/v3.1.1/ (Accessed: 20 May 2021).

ETSI (2021b) *SAREF repository*. Available at: https://forge.etsi.org/rep/SAREF (Accessed: 20 May 2021).

Feng, S. C., Kumaraguru, S. and Sun, X. (2015) 'Energy Assessment Methodology for Mechanical and Electrical Product Assembly Processes'. doi: 10.1115/MSEC2015-9378.

Fraunhofer IWU. (2021) *KOMMA - RDF Object Mapper and Editing Framework for Java*. Available at: https://komma.enilink.net/ (Accessed: 20 May 2021).

International Organization for Standardization (ISO) (2004) *ISO 10303-11:2004 Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*. Available at: https://www.iso.org/standard/38047.html.

Kim, K. Y., Manley, D. G. and Yang, H. (2006) 'Ontology-based assembly design and information sharing for collaborative product development', *CAD Computer Aided Design*, 38(12), pp. 1233–1250. doi: 10.1016/j.cad.2006.08.004.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

Lambda Architecture » λ lambda-architecture.net (no date). Available at: http://lambda-architecture.net/ (Accessed: 18 May 2021).

Lee W., L. (2006) *Interchanging discrete event simulation process-interaction models using the web ontology language - OWL.* University of Central Florida. Available at: https://www.researchgate.net/publication/221527552_Interchanging_discrete_event_simulation_process-interaction_models_using_the_web_ontology_language_-_OWL (Accessed: 7 June 2021).

de Leeuw, V. (2019) *Concepts and Applications of the I4.0 Asset Administration Shell*, *ARC Advisory Group blog article*. Available at: https://www.arcweb.com/blog/concepts-applications-i40-asset-administration-shell (Accessed: 7 June 2021).

Lemaignan, S. *et al.* (2006) 'MASON: A proposal for an ontology of manufacturing domain', *Proceedings - DIS 2006: IEEE Workshop on Distributed Intelligent Systems - Collective Intelligence and Its Applications*, 2006, pp. 195–200. doi: 10.1109/DIS.2006.48.

Li, G., Zhang, L. and Gao, Q. (2010) 'A virtual assembly-oriented multi-views information model and its XML description', in *Chinese Control and Decision Conference*. Xuzhou, pp. 1294–1298. doi: 10.1109/CCDC.2010.5498176.

Lohse, N. (2006) *Towards an ontology framework for the integrated design of modular assembly systems*, *University of Nottingham*. Available at: http://etheses.nottingham.ac.uk/1446/.

Louvel, M. and Pacull, F. (2014) 'LINC: A compact yet powerful coordination environment', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp. 83–98. doi: 10.1007/978-3-662-43376-8_6.

Miller, J. A. *et al.* (2004) 'Investigating ontologies for simulation modeling', in *Proceedings of the IEEE Annual Simulation Symposium*, pp. 55–63. doi: 10.1109/simsym.2004.1299465.

MLOps: Continuous delivery and automation pipelines in machine learning (no date). Available at: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (Accessed: 28 May 2021).

Modelica Association (2020) *Functional Mock-up InterfaceforModel Exchange and Co-Simulation*. Available at: https://github.com/modelica/fmi-standard/releases/download/v2.0.2/FMI-Specification-2.0.2.pdf.

MQTT Client and Broker and MQTT Server and Connection Establishment Explained - MQTT Essentials: Part 3 (no date). Available at: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/ (Accessed: 24 May 2021).

Object Management Group (2017) *Unified Modeling Language, version 2.5.1*. Available at: https://www.omg.org/spec/UML/2.5.1/PDF.

Object Management Group (2019a) *OMG Meta Object Facility (MOF) Core Specification*. Available at: https://www.omg.org/spec/MOF/2.5.1/PDF.

Object Management Group (2019b) *Systems Modeling Language, version 1.6*. Available at: https://www.omg.org/spec/SysML/1.6/PDF.

OSIsoft | Operational Intelligence | PI System (no date). Available at: https://www.osisoft.com/ (Accessed: 29 June 2021).

Pacull, F. (2017) 'Because it is not only about connecting objects', *Revue Genie Logiciel*, March.

Plattform Industrie 4.0 (2019) *Details of the Asset Administration Shell - Part 1-The exchange of information between partners in the value chain of Industrie 4.0 (Version 2.0.1)*. Available at: https://www.plattform-

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V2.pdf.

Plattform Industrie 4.0 (2020) *Details of the Asset Administration Shell Part 2 – Interoperability at Runtime – Exchanging Information via Application*. Available at: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.pdf.

Rachuri, S. *et al.* (2006) 'A model for capturing product assembly information', *Journal of Computing and Information Science in Engineering*, 6(1), pp. 11–21. doi: 10.1115/1.2164451.

RDF Schema 1.1 (2014). Available at: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Sylla, A. N., Louvel, M. and Rutten, É. (2016) 'Combining transactional and behavioural reliability in adaptive middleware', in *ARM 2016 - 15th Workshop on Adaptive and Reflective Middleware, colocated with ACM/IFIP/USENIX Middleware 2016*. Association for Computing Machinery, Inc. doi: 10.1145/3008167.3008172.

Teo, Y. M. and Szabo, C. (2008) 'CODES: An integrated approach to composable modeling and simulation', in *Proceedings - Simulation Symposium*, pp. 103–110. doi: 10.1109/ANSS-41.2008.24.

The Linux Foundation (2021) *Open Neural Network Exchange - The open standard for machine learning interoperability*. Available at: https://onnx.ai/ (Accessed: 18 May 2021).

TopQuadrant, I. (2021) *TopBraid Composer – Maestro Edition*. Available at: https://www.topquadrant.com/products/topbraid-composer/ (Accessed: 26 May 2021).

Turnitsa, C. and Tolk, A. (2005) 'Evaluation of the C2IEDM as an interoperability-enabling Ontology', *European Simulation Interoperability Workshop 2005*, (January 2005), pp. 356–368.

University Standford (2021) *Protégé - A free, open-source ontology editor and framework for building intelligent systems*. Available at: https://protege.stanford.edu/ (Last Accessed: 26 May 2021).

Wenzel, K. *et al.* (2011) 'Semantic Web based dynamic energy analysis and forecasts in manufacturing engineering', *Glocalized Solutions for Sustainability in Manufacturing - Proceedings of the 18th CIRP International Conference on Life Cycle Engineering*, pp. 507–512. doi: 10.1007/978-3-642-19692-8_88.

What are the 4 Vs of Big Data? . Available at: https://bernardmarr.com/default.asp?contentID=2119 (Last Accessed: 18 May 2021).

What is Data Architecture? How to Drive Real Business Results through Data. Available at: https://www.talend.com/resources/what-is-data-architecture/ (Last Accessed: 17 June 2021).

World Wide Web Consortium (W3C) (2004a) *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Available at: https://www.w3.org/Submission/SWRL/.

World Wide Web Consortium (W3C) (2004b) *XML Schema Part 0: Primer Second Edition*. Available at: https://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

World Wide Web Consortium (W3C) (2006) *Extensible Markup Language (XML) 1.1 (Second Edition)*. Available at: https://www.w3.org/TR/2006/REC-xml11-20060816/.

World Wide Web Consortium (W3C) (2008) *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available at: https://www.w3.org/TR/2008/REC-xml-20081126/.

World Wide Web Consortium (W3C) (2012a) *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Available at: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

World Wide Web Consortium (W3C) (2012b) *OWL 2 Web Ontology Language Primer (Second Edition)*. Available at: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.

World Wide Web Consortium (W3C) (2012c) *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*. Available at: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

World Wide Web Consortium (W3C) (2012d) *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. Available at: https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/.

World Wide Web Consortium (W3C) (2012e) *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. Available at: https://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/.

World Wide Web Consortium (W3C) (2013a) *RIF Overview (Second Edition)*. Available at: https://www.w3.org/TR/rif-overview/.

World Wide Web Consortium (W3C) (2013b) *RIF Primer (Second Edition)*. Available at: https://www.w3.org/TR/2013/NOTE-rif-primer-20130205/.

World Wide Web Consortium (W3C) (2013c) *SPARQL 1.1 Overview*. Available at: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

World Wide Web Consortium (W3C) (2013d) *SPARQL 1.1 Query Language*. Available at: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

World Wide Web Consortium (W3C) (2014a) *RDF 1.1 Concepts and Abstract Syntax*. Available at: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

World Wide Web Consortium (W3C) (2014b) *RDF 1.1 Primer*. Available at: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

World Wide Web Consortium (W3C) (2014c) *RDF 1.1 Semantics*. Available at: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

World Wide Web Consortium (W3C) (2017a) *SHACL Use Cases and Requirements*. Available at: https://www.w3.org/TR/2017/NOTE-shacl-ucr-20170720/.

World Wide Web Consortium (W3C) (2017b) *Shapes Constraint Language (SHACL)*. Available at: https://www.w3.org/TR/2017/REC-shacl-20170720/.

World Wide Web Consortium (W3C) (2017c) *XSL Transformations (XSLT) Version 3.0*. Available at: https://www.w3.org/TR/xslt-30/.

World Wide Web Consortium (W3C) (2021a) *W3C Data Activity - Building the Web of Data*. Available at: https://www.w3.org/2013/data/ (Accessed: 26 May 2021).

World Wide Web Consortium (W3C) (2021b) *Web of Things - Documentation*. Available at: https://www.w3.org/WoT/documentation/ (Accessed: 26 May 2021).

Zeigler, B. P., Praehofer, H. and Kim, T. G. (2000) *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958339.