D2.1 Common Data Models and Semantic Interoperability Mechanisms - Release 1

Dissemination Level: Public
Submission Date: 18/05/2020

Table of Contents

List of Figures

List of Tables

1 Executive Summary

DEMETER aims to lead the Digital Transformation of the European Agrifood sector based on the rapid adoption of advanced technologies, such as Internet of Things, Artificial Intelligence, Big Data, Decision Support, Benchmarking, Earth Observation, etc., in order to increase performance in multiple aspects of farming operations, as well as to assure the viability and sustainability of the sector in the long term. It aims to put these digital technologies at the service of farmers using a human-in-the-loop approach that constantly focuses on mixing human knowledge and expertise with digital information. DEMETER focuses on interoperability as the main digital enabler, extending the coverage of interoperability across data, platforms, services, applications and online intelligence, as well as human knowledge, and the implementation of interoperability by connecting farmers and advisors with providers of ICT solutions and machinery.

To enable the achievement of the aforementioned objectives, and to promote the targeted technological, business, adoption and socio-economic impacts, DEMETER has already delivered a Reference Architecture (RA) that is suitable to address these challenges in the agrifood domain. In order to implement this Reference Architecture several key technologies need to be developed. The most crucial of these is the common data models which make the DEMETER Agricultural Information Model (AIM) and which enable semantic interoperability between DEMETER and existing agrifood systems and ontologies. This deliverable introduces the initial release of the DEMETER common data models and interoperability mechanisms that will be used in the development of the DEMETER enabled apps for the first round of the DEMETER pilots. This data model specification follows the analysis of the State of the Art and the first elicitation of the requirements regarding the data that it needs to model. It initially discusses and reviews the State of the Art regarding data models and related ontologies which are consulted in order to develop the DEMETER AIM. Afterwards, it presents the requirements that drive the development of the DEMETER AIM. It then describes in detail the initial design and development of the DEMETER Agricultural Information Model and its core metamodel; then the cross-domain ontology, followed by the domain specific ontologies; and at last the metadata schema used by AIM. Following this description, it discusses the interoperability support between the DEMETER AIM and several existing ontologies and dominant agri-food systems by detailing the semantic mapping of those to AIM. Afterwards, it presents the AIM implementation which follows a layered and modular approach, reusing as much as possible existing ontologies and vocabularies. Finally, the document concludes presenting also future work towards the final version of the AIM.

2 Acronyms

ADAPT	Agricultural Data Application Programming Toolkit
AEF	Agricultural Industry Electronics Foundation
AF	Aquaculture Facilities
AgroRDF	Agriculture Resource Description Framework
AgroXML	XML dialect for representing and describing farm work
AIM	Agricultural Information Model
AKIS	Agricultural Knowledge and Innovation Systems
API	Application Programming Interface
BT	Broader thesaurus
CAN-Bus	Controller Area Networks Bus
CIM	Context Information Management
CO2	Carbon Dioxide
DCAT	Data Catalog Vocabulary
DDL	Description Definition Language
DG	Directorate-Generals
DQV	Data Quality Vocabulary
DSS	Decision Support System
DUV	Data Usage Vocabulary
EA	Enterprise Architect
EO	Earth Observation
ESRI	Environmental Systems Research Institute
ETL	Extract-Transform-Load
ETSI	European Telecommunications Standards Institute
EU	European Union
EVI	Enhanced Vegetation Index

FAIR	Findable, Accessible, Interoperable and Reusable
FAO	Food and Agriculture Organization
FMIS	Farm Management Information System
GeoJSON	Geographical JavaScript Object Notation
GeoSPARQL	Geographic Simple Protocol and RDF Query Language
GEOSS	Global Earth Observation System of Systems
GML	Geography Markup Language
GPS	Global Positioning System
GSA	Global Navigation Satellite Systems Agency
GSMA	Global System Mobile Association
HTTP	HyperText Transfer Protocol
ICAR	International Committee for Animal Recording
ICT	Information and communications technology
IDS	International Data Spaces
IDSA	International Data Spaces Association
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMA	Information Management Adapter
IoT	Internet of Things
IR	Implementing Rules
ISG	Industry Specification Group
ISO	International Organization for Standardisation
ISO-XML	International Organization for Standardisation - Extensible Markup Language
IT	Information Technology
JRC	Joint Research Centre
JSON	Java Script Object Notation

JSON-LD	Java Script Object Notation - Linked Data
KB	Knowledge Base
KTBL	Kuratorium für Technik und Bauwesen in der Landwirtschaft (Board of Trustees for Technology and Construction in Agriculture)
LOD	Linked Open Data
MQTT	Message Queuing Telemetry Transport
MUTO	Modular Unified Tagging Ontology
NDMI	Normalized Difference Moisture Index
NDRE	Normalized Difference Red Edge
NDVI	Normalized Difference Vegetation Index
NGSI	Next Generation Sensors Initiative
NGSI-LD	Next Generation Sensors Initiative - Linked Data
NGSIv2	Next Generation Services Interface v2
NoSQL	No Structured Query Language
NT	Narrower thesaurus
O&M	Observation and Measurements
OBO	Open Biological and Biomedical Ontology
OCB	Orion Context Broker
ODRL	W3C Open Digital Rights Language
OGC	Open Geospatial Consortium
OMA	Open Mobile Alliance
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logic
OWL-RL	Web Ontology Rule Language
PROV-O	The Provenance Ontology
pySHACL	Python Shapes Constraint Language

QB	RDF Data Cube Vocabulary
QUDT	Quantities, Units, Dimensions, and Types Ontology
RA	Reference Architecture
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RFID	Radio Frequency IDentification
SAREF	Smart Appliances REFerence (SAREF) ontology
Saref4Agri	Smart Appliances REFerence (SAREF) ontology for agriculture
ScienceDMZ	Science Demilitarized Zone
SDMX	Statistical Data and Metadata eXchange
SDO	Standards Development Organization
SDWWG	Spatial Data on the Web Working Group
SHACL	Shapes Constraint Language
SKOS	Simple Knowledge Organization System
SKOS-XL	Simple Knowledge Organization System eXtension for Labels
SOS	Sensor Observation Services
SOSA	Sensor, Observation, Sample, and Actuator
SPADE	Standardized Precision Ag Data Exchange
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
SSN	Semantic Sensor Network
SWRL	Semantic Web Rule Language
TC23	Tractors and machinery for agriculture and forestry
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
US	United States
UTC	Coordinated Universal Time (Temps Universel Coordonné)
W3C	World Wide Web Consortium
W3C-TR	World Wide Web Consortium Technical Report
WCS	Web Coverage Service
WGS84	World Geodetic System 1984
WoT	Web of Things
WP	Work Package
WSDL	Web Services Description Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

3 List of Authors and Reviewers

Organisation	Author
ICCS	Ioanna Roussaki (Editor)
	Ioannis Vetsikas
	George Routis
	Marios Paraskevopoulos
PSNC	Raul Palma
	Soumya Brahma
	Szymon Mueller
Fraunhofer FIT	Till Döhmen
	Christoph Lange
	Johannes Lipp
OGCE	Rob Atkinson
ENG	Antonio Caruso
	Angelo Marguglio
TECNALIA	Sonia Bilbao
	Belén Martínez
	Alejandro Rodríguez
	Fernando Jorge

Organisation	Reviewer
Fraunhofer IESE	Patricia Kelbert
	Anna Maria Vollmer
UMU	Manuel Mora
OdinS	Juan Antonio Martínez
Vicomtech	Óscar Miguel Hurtado
DNET Labs	Nenad Gligoric
TSSG	Nithin Padmaprabhu

4 Introduction

This deliverable presents the first release of the "DEMETER Common Data Models and Semantic Interoperability Mechanisms". It also presents the DEMETER Agricultural Information Model (AIM) which will be the data model used by all partners for the first round of the DEMETER pilots. This data model builds upon a thorough analysis of both the related State of the Art and state of the practice. It is guided by the initial elicited requirements and, of course, by the DEMETER vision and targeted objectives.

More specifically, the rest of the document is structured as follows:

Section 5 provides an analysis of the state of the art (and state of the practice) on related data models and interoperability mechanism systems that exist in the domain of Smart Agrifood. These are agrifood Data Models, mechanisms and systems, that promote semantic interoperability in this domain. We initially present general data models that are commonly used, then we present specific ontologies (e.g. Saref4agri) and last we present mechanisms that allow semantic interoperability. This analysis, together with the technical requirements (presented in the following section), drives the design and development of the DEMETER AIM.

Section 6 gives an overview of the technical requirements extracted by Task 2.1 (T2.1). This is an exhaustive list of specific technical requirements that the DEMETER Agricultural Information Model needs to be able to represent, as well as requirements regarding the interoperability with existing systems and ontologies, including the mapping of these data models to the DEMETER AIM.

Section 7 presents the initial development of the DEMETER Agricultural Information Model (AIM) in detail and follows a modular approach in a layered architecture for its development. More specifically:

- subsection 7.1 presents its core metamodel, which follows the NGSI-LD meta-modeling approach;
- subsection 7.2 presents the cross-domain ontology used, i.e. the set of generic models which aim at providing common definitions for all agrifood domain handled by the AIM and at avoiding conflicting or redundant definitions of the same classes at the domain-specific layer;
- subsection 7.3 presents the domain specific ontologies developed for the AIM, which model information such as crops, animals, agricultural products as well as farms and farmers, etc.
- subsection 7.4 describes the metadata schema used by the AIM. It expresses semantics, related to meta-information about the datasets, based on the cross-domain and domain specific ontologies previously presented.

Section 8 presents the interoperability support between the DEMETER AIM and several existing ontologies and dominant agri-food systems detailing the semantic mapping of these to the AIM. More specifically, it presents this information for the following dominant systems: FIWARE AgriFood, Saref4Agri, ADAPT, INSPIRE (and FOODIE), AGROVOC and EO data.

Section 9 presents the implementation of the DEMETER AIM which follows a layered and modular approach,

reusing as much as possible existing ontologies and vocabularies. More specifically, it presents the implementations for the different layers (parts) of AIM, together with the design and implementation choices taken, the mappings implemented and the tools used during the implementation process.

Finally, Section 10 concludes the document presenting also future work towards the final version of the AIM (delivarable D2.3), while Section 11 provides the respective references used and Annex A records the expected input data engaged in the 20 DEMETER pilots, as these have been reported by WP5.

The models and interoperability mechanisms presented in this deliverable complement the deliverable D3.1 DEMETER Reference Architecture (Release 1). This work will be used in the following deliverables which follow:

- D2.2 DEMETER data and knowledge extraction tools (May/June 2020)
- D3.2 DEMETER technology integration tools (June 2020)
- D4.1 Decision Support, Benchmarking and Performance Indicator Monitoring Tools – Release 1 (May 2020)
- D4.2 Decision Enablers, Advisory Support Tools and DEMETER Stakeholder Open Collaboration Space (June 2020)
- D5.3 Testbed, deployment, system extensions and applications for pilot round 1 (July 2020)

The revised (final) version of the DEMETER Common Data Models and Semantic Interoperability Mechanisms is planned for release on April 2021 and will be presented in D2.3.

5 State of the Art

This section presents the State of the Art for Task 2.1 Common Data Models and Semantic Interoperability of the DEMETER project. First, we present general data models that are commonly used, then we present specific ontologies (e.g. Saref4agri, or a specialized one for weather data or sensors, etc.) and, last, we present mechanisms that allow semantic interoperability.

5.1 FIWARE NGSI

NGSI is a protocol developed to manage Context Information. The context of an entity consists of:

- a set of characteristics that describe it, including its (dynamic) state;
- other entities with which it has defined relationships, and the nature of those relationships.

We can define Context Information as the informational representation of a context as defined above. We define property as a description instance, which associates a value, to either an entity, a relationship or another property. For instance, the terms “speed”, “soilTemperature”, “windDirection”. A relationship describes the conceptual connection from one entity to another entity in a context, for example "adjacent to", "owned by", "created by", etc.

A Context Information Management (CIM) is a platform or system (usually named Context Broker) which provides the following services: *context information registry, discovery, publication, mediation, modification or notification*. Cross-cutting context information management provides CIM between independent target domains (yet obviously can also handle same or similar domains).

What a CIM does is to collect information from user-driven applications, platforms managing end-devices and other sources to provide it to applications via a CIM API. The CIM system enables use-cases which link together disparate but related information. IoT services will be enhanced when applications have access to a full set of context information. A CIM system potentially enriches services by bringing together information from a wider set of service-relevant sources that would otherwise be unavailable.

In the Smart Agrifood domain (for which we will present specific data models in a later section of this report), context information can be composed by the state of entities such as drones, tractors, greenhouses, parcels, etc. Also, other other context information might be relevant and can be offered by specialized services or databases, for instance weather forecasts, pests or diseases historical information, etc.

The context/Data Broker middleware can offer application interfaces for CIM so that there is a real time view of what is happening. The sections below describe what standard application interfaces are available for CIM and related implementations.

FIWARE NGSI is an instantiation (binding) of the OMA NGSI-9 and NGSI-10 abstract interfaces for Context

Information Management. Version 2 of the FIWARE binding (*FIWARE NGSIv2*) is based on JavaScript Object Notation (JSON) and HTTP/REST and follows the usual, de-facto industry standards. Such API binding for CIM has been recommended by the GSMA for IoT and Big Data Projects.

NGSI can support the representation of context information, with a meta-model based on entities, attributes and extra attribute's metadata, which is well-known and powerful approach usually extended to the NGSI-LD format (see next subsection for NGSI-LD). Attributes can represent the properties of an entity or can point to (using a priori defined conventions) other entities. Experience shows that instantiations of this meta-model can be easily mapped/implemented using a wide variety of data stores, including NoSQL, SQL or even Graph Databases. Furthermore, the NGSI information meta-model is quite close to other meta-models widely used in the industry, namely schema.org, thus enabling interoperability and reusability.

Figure 1 shows the general framework of a Context Information Management (CIM) system for the Agrifood domain with the corresponding Context Broker, while Figure 2 presents a specific example of the information that might be stored in such a Context Broker. For more use cases of CIM see [ETS18].

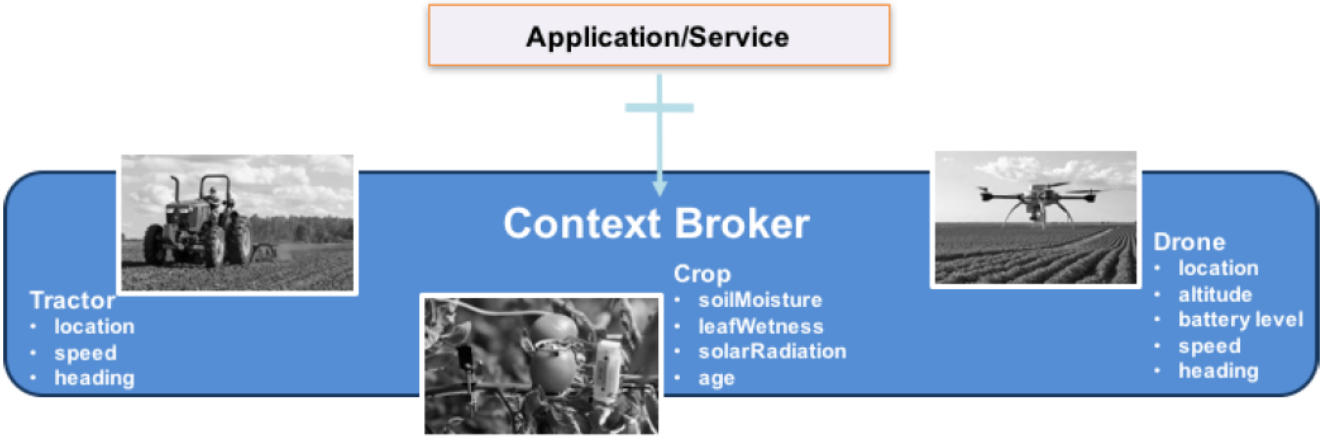


Figure 1. Example Context Broker framework for an Agrifood system

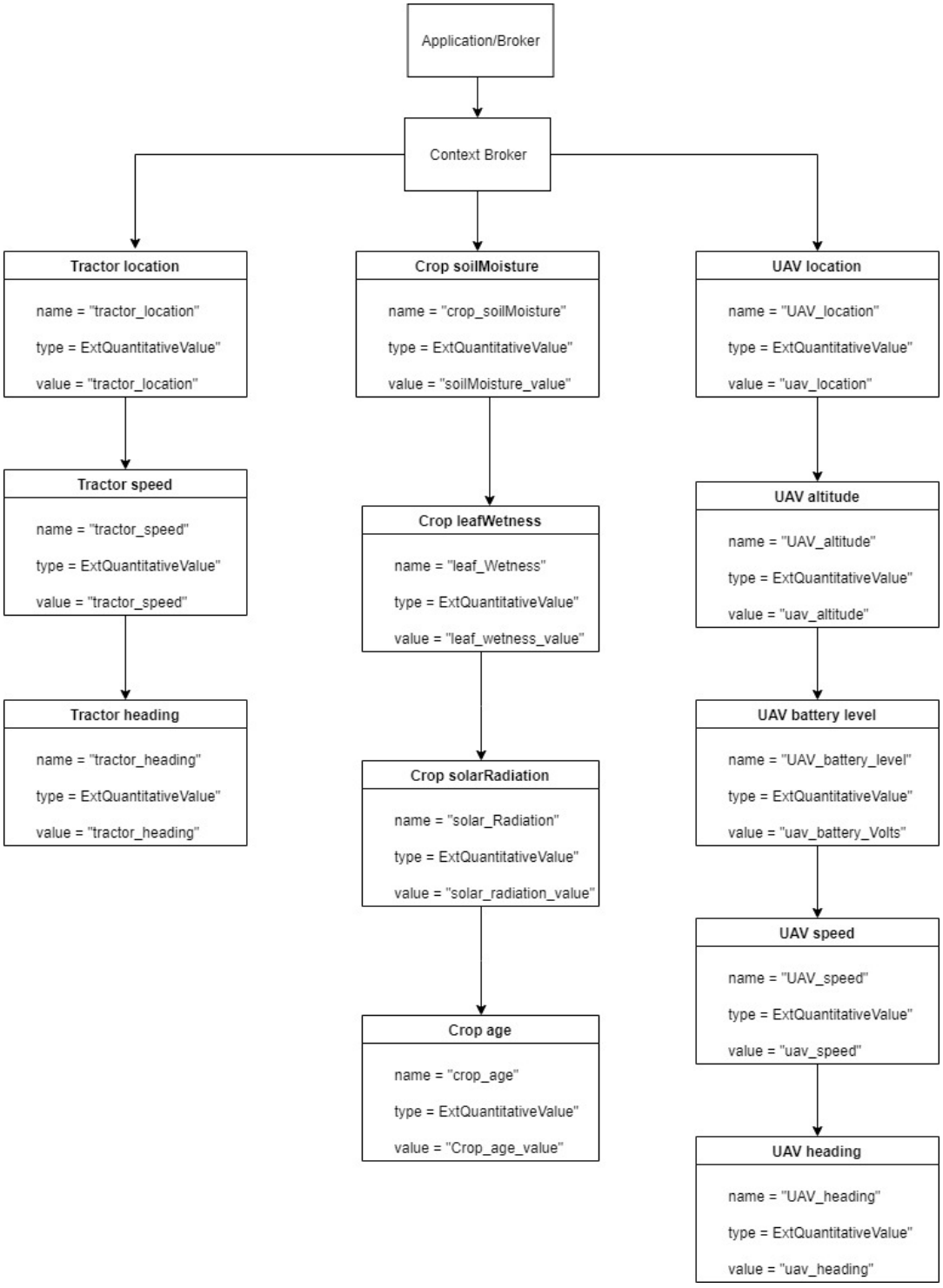


Figure 2. Example of the information stored in an Agrifood System Context Broker

5.1.1 The NGSI meta-model

Figure 3 below depicts the NGSI meta-model. Central to this model are entities in the NGSI information model, each with an Entity Id. The type system of NGSI enables entities to have an Entity Type. Entity Types are semantic types, which means that they are intended to describe the type of thing represented by the entity. For example, a context entity with id “tractor-128” could have the type “Tractor”. Each entity is uniquely identified by the combination of its id and type. It is noteworthy that these elements are always mandatory when the meta-model is instantiated.

Attributes are properties of context entities. For example, the current speed of a tractor could be modelled as attribute “speed” of an entity “tractor-30”. In the NGSI data model, attributes have an attribute name, an attribute type, an attribute value and metadata. The attribute name describes what kind of property the attribute value represents of the entity, for example “speed”. The attribute type represents the NGSI value type of the attribute value, which is usually similar or equivalent to the JSON data type. The attribute value, finally, contains the actual data and optional metadata describing properties of the attribute value, e.g. accuracy, provider, or an observation timestamp.

Metadata is used as an optional part of the attribute value as described above. Similar to attributes, each piece of metadata has a metadata name, describing the role of the metadata in the place where it occurs; for example, the metadata name “accuracy” indicates that the metadata value describes how accurate a given attribute value is; a metadata type describes the NGSI value type of the metadata value; a metadata value contains the actual metadata.

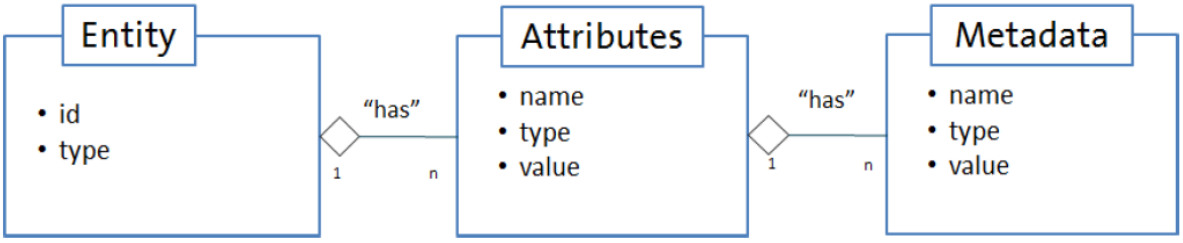


Figure 3. NGSI meta-model

5.2 NGSI-LD

The OMA NGSI information model is currently being evolved as an evolution of NGSI to better support linked data (entity’s relationships), property graphs and semantics (thus exploiting the capabilities offered by JSON-LD).

This work is being conducted under the ETSI ISG CIM initiative[1] and has been branded as NGSI-LD [NGS19]. It is noteworthy that the ETSI ISG CIM information model is a generalization of the existing OMA NGSI information model. As a result, a good level of compatibility and a clear migration path between both information models is to be expected. While this still seems yet work-in-progress, this model offers a range of new possibilities and therefore a clearer view of the roadmap should be provided.

The NGSI-LD Information Model [ETS18] prescribes the structure of context information that shall be supported by an NGSI-LD system. It specifies the data representation mechanisms that shall be used by the NGSI-LD API itself. In addition, it specifies the structure of the Context Information Management vocabularies to be used in conjunction with the API.

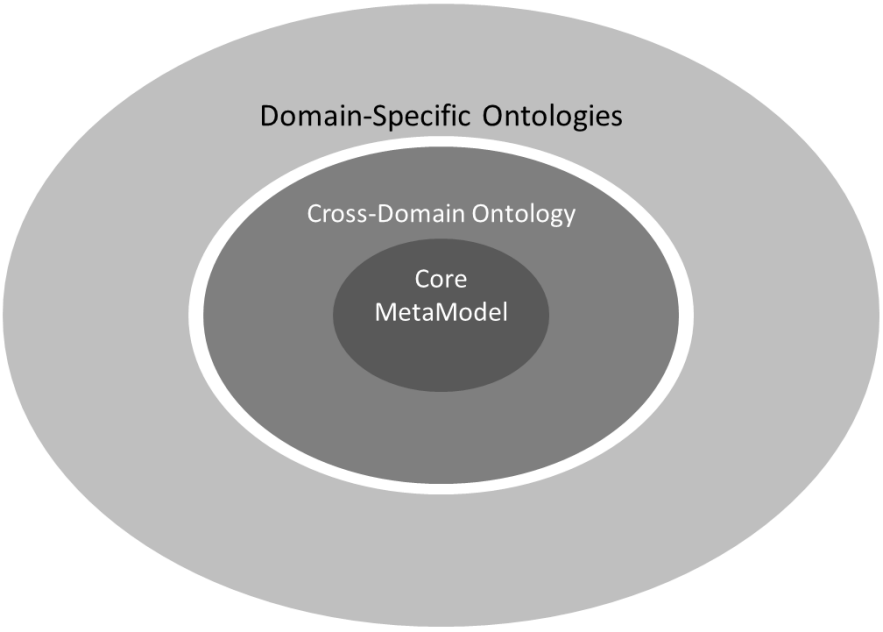


Figure 4. Overview of the NGSI-LD Information Model Structure

The NGSI-LD Information Model is defined at two levels (see Figure 4): the foundation classes, which correspond to the Core Meta-model, and the Cross-Domain Ontology. The former amounts to a formal specification of the "property graph" model. The latter is a set of generic, transversal classes which are aimed at avoiding conflicting or redundant definitions of the same classes in each of the domain-specific ontologies. Below these two levels, domain-specific ontologies or vocabularies can be devised. For instance, the SAREF Ontology[2] can be mapped to

[1] An one page summary of (the context information manager) NGSI-LD as proposed to W3C can be found in: https://docbox.etsi.org/ISG/CIM/Open/One_page_summary_NGSI-LD_W3C_Workshop_Graph_Data.pdf.
[2] The Shared Appliances REFerence (SAREF) ontology is described in https://ontology.tno.nl/saref/. A number of SAREF extensions can be found in http://saref.linkeddata.es/.

the NGSI-LD Information Model, so that, for example, smart home applications may benefit from this Context Information Management API specification.

The version of the cross-domain model proposed by the present document is a minimal one, aimed at merely defining the classes used in this release of the API specification. It will be extended in later versions with classes defining extra concepts such as mobile vs. fixed entities, state properties vs. instantaneous vs. fixed properties, etc.

5.2.1 NGSI-LD Meta Model

Figure 5 provides a graphical representation of the NGSI-LD Meta-Model in terms of classes and their relationships. To provide additional clarity, an informal (non-normative) mapping to the Property Graph Model is also presented.

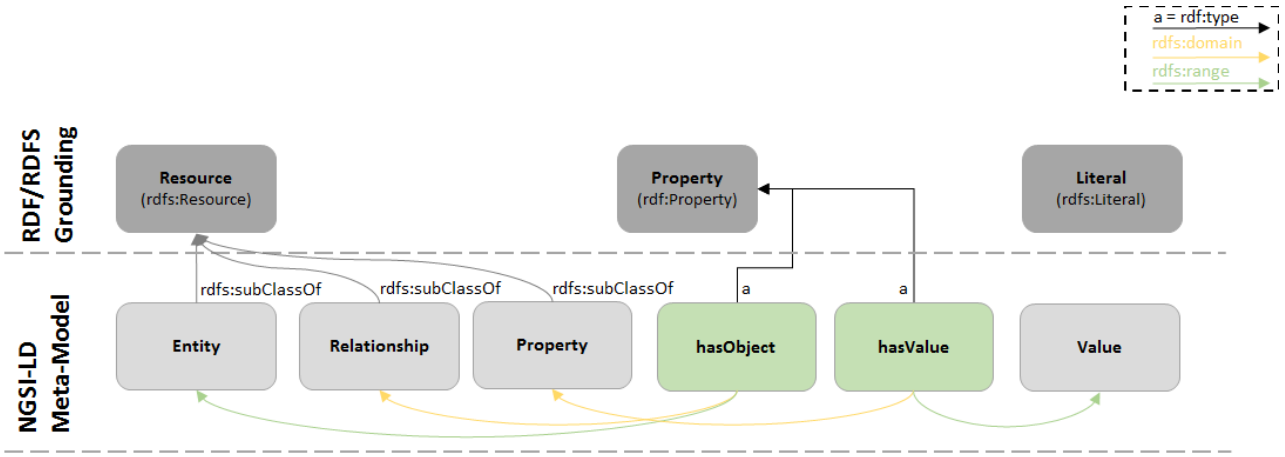


Figure 5. NGSI-LD Core Meta-Model

Implementations shall support the NGSI-LD Meta-model as follows:[3]

- An **NGSI-LD Entity** is a subclass of rdfs:Resource.
- An **NGSI-LD Relationship** is a subclass of rdfs:Resource.
- An **NGSI-LD Property** is a subclass of rdfs:Resource.
- An **NGSI-LD Value** shall be either an rdfs:Literal or a node object (in JSON-LD language) to represent complex data structures.
- An **NGSI-LD Property** shall have a **value,** stated through *hasValue*, which is of type rdf:Property.

[3] This information is taken as is from [ETS18]. RDF stands for Resource Description Framework and more details about it can be found in [RDF14].

- An **NGSI-LD Relationship** shall have an **object** stated through *hasObject*, which is of type rdf:Property.

5.2.2 NGSI-LD Representation

In this section, we show how the NGSI-LD information model could be described using UML diagrams. This is done in order to help those readers less familiar with ontology representations or RDF [RDF14] to understand the NGSI-LD Information Model.

Figure 6 below shows a UML diagram which describes the ETSI ISG CIM information model.[ETS18] The RDF's Relationship, Property and Value are represented as UML classes in this diagram. UML associations are used to interrelate these classes while keeping the structure and semantics defined by the NGSI-LD Information Model.

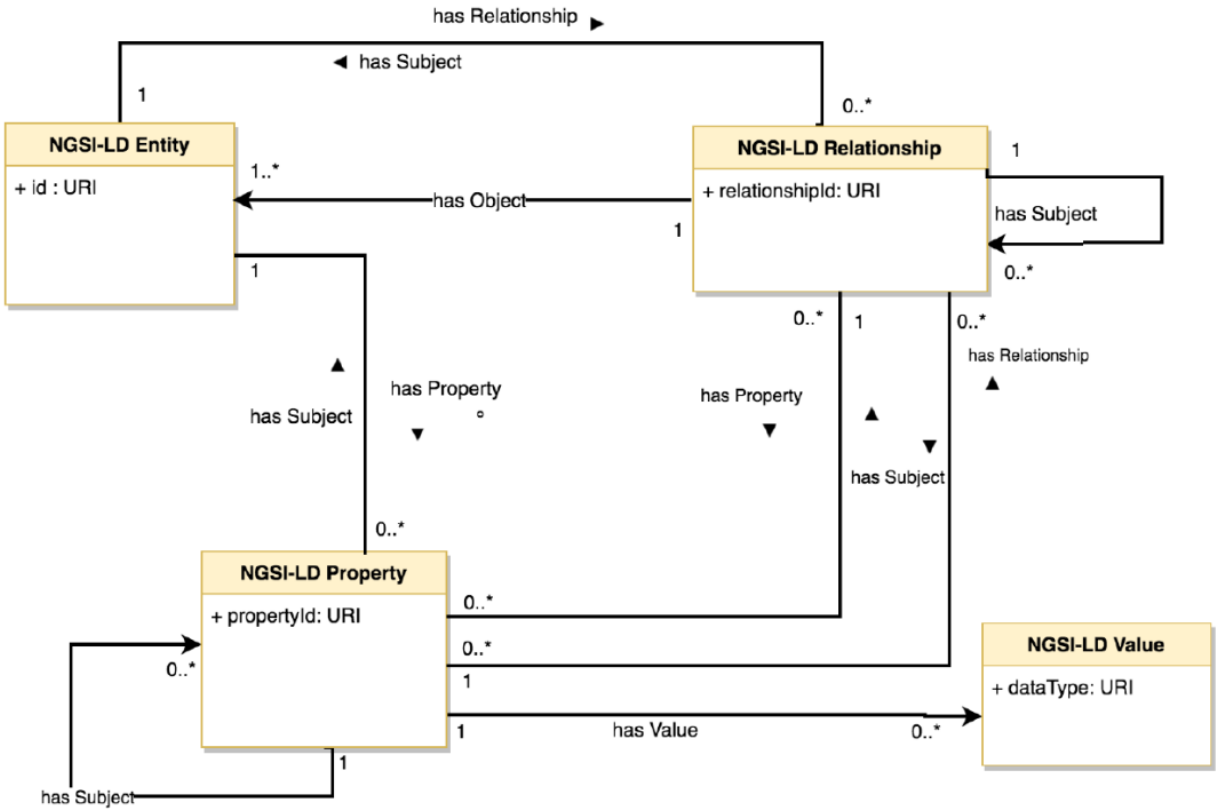


Figure 6. ETSI NGSI-LD information model as UML diagram

The main constructs are NGSI-LD Entity, NGSI-LD Property and NGSI-LD Relationship. NGSI-LD Entities (instances) can be the subject of NGSI-LD Properties or NGSI-LD Relationships. In terms of the traditional NGSI data model, NGSI-LD Properties are the combination of an attribute (property) and its value. NGSI-LD Relationships allow

programmers to establish relationships between instances using linked data. In practice, they are NGSI attributes, but with a special value which happens to be a URI which points to another entity (internally or externally). They are similar to the "ref" attributes already mentioned. NGSI-LD Properties and NGSI-LD Relationships can be the subject of other NGSI-LD Properties or NGSI-LD Relationships. Thus, in the ETSI ISG CIM information model, there are no attribute's metadata but just "properties of properties". It is not expected to have infinite graphs and, in practice, only one or two levels of property or relationship "chaining" will happen. Usually, there will be only one, equivalent to the NGSI metadata abstraction. NGSI-LD Entities are represented using JSON-LD, a JSON-based serialization format for Linked Data. The main advantage of JSON-LD, apart from the ability to represent linked data, is that it offers the capability of expanding JSON terms to URIs so that vocabularies can define terms unambiguously.

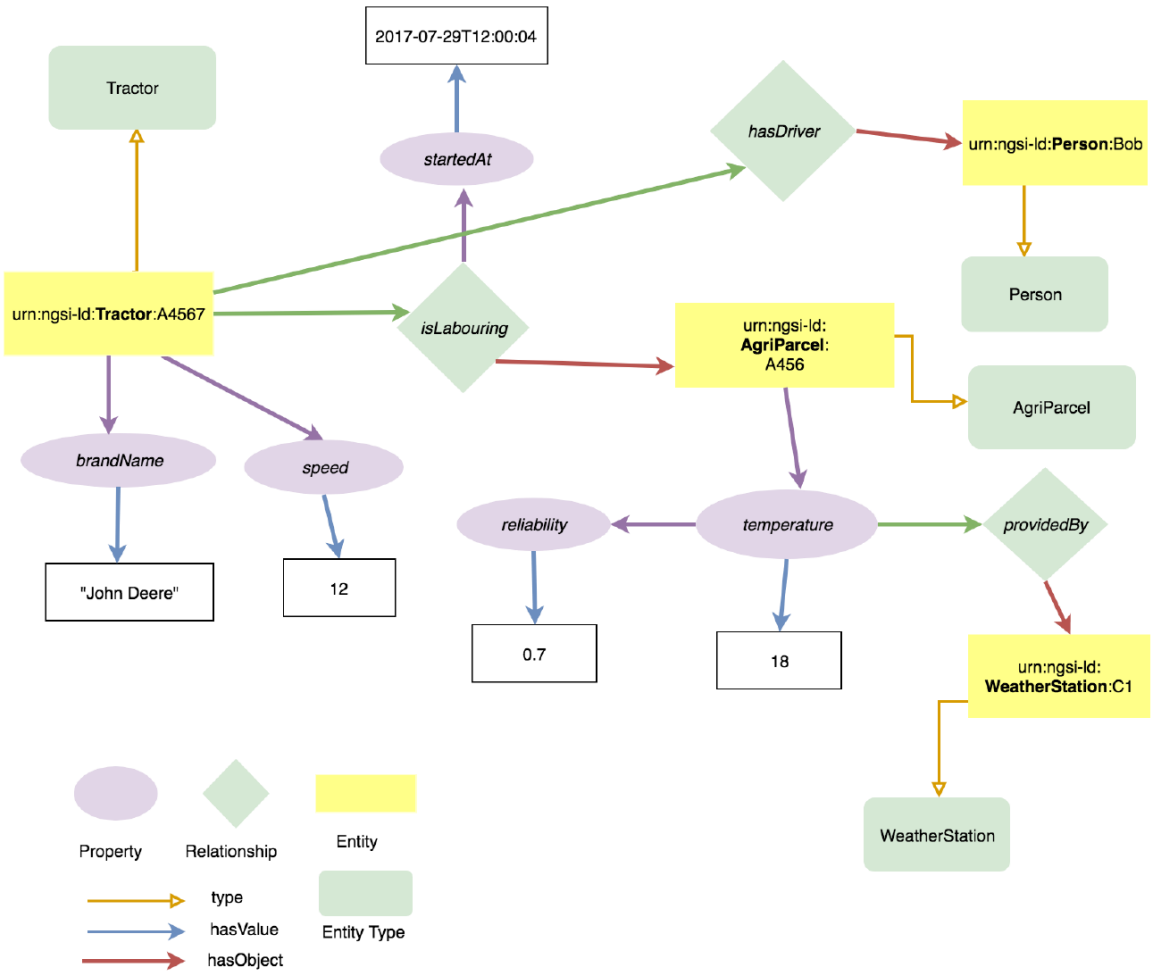


Figure 7. Smart Agrifood ETSI ISG CIM Information Model

Figure 7 shows an instantiation example of this information model pertaining to Smart Agrifood.[4] It conveys that there is an instance of an entity of type “Tractor” whose current driver is a person (entity type “Person”). The tractor is performing a task in a parcel (entity type “AgriParcel”). Different properties about those entities are provided (“speed, “brandName”, “temperature”, etc.) and additional properties of properties (for instance, an accuracy level) or properties of relationships (“startedAt”) are described.

We now give an example of the JSON-LD serialization corresponding to some of the “entities” represented in Figure 7. NGSI-LD Entities are represented as JSON-LD objects, which are regular JSON objects that incorporate a special tagged member (named “@context”), whose value provides the mapping between terms (short-hand strings) and fully qualified names (URIs), so that every term in the JSON object is unambiguously identified.

```
"id": "urn:ngsi-ld:Tractor:A4567",
"type": "Tractor",
"brandName":
{
    "type": "Property",
   "value": "John Deere"
},
"isLabouring":
{
    "type": "Relationship",
    "object": "urn:ngsi-ld:AgriParcel:A456",
    "startedAt":
     {
         "type": "Property",
         "value": "2017-07-29T12:00:04",
     }
   },
"hasDriver":
{
 "type": "Relationship",
 "object": "urn:ngsi-ld:Person:Bob"
},
```

[4] It is taken from [IOF18], but has also been presented elsewhere, e.g. https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-how-to-design-datamodels.

```
"@context": http://example.org/agri/iof2020/context.jsonld"
{
 "id": "urn:ngsi-ld:AgriParcel:A456",
 "type": "AgriParcel",
 "temperature":
 {
      "type": "Property",
      "value": 18,
      "reliability":
      {
          "type": "Property",
          "value": 0.7
      },
      "providedBy":
      {
          "type": "Relationship",
          "object": "urn:ngsi-ld:WeatherStation:C1"
      },
      "@context": "http://example.org/agri/iof2020/context.jsonld"
   }
}
```

Below, in Figure 8, we can see an example of the JSON-LD representation defined by NGSI-LD [LFr19]. In this case, we have an entity whose type is Vehicle, which has a property (here the brand name), and a relationship. This vehicle has been involved in an accident ("inAccident") crashing into a lamppost (another entity). This relationship, in turn, is also related to the officer, Officer123, who presented this fact.

Figure 8. Example of NGSI-LD representation based on JSON-LD

5.2.3 NGSI-LD API

To express the example of Figure 8, it is necessary to use the NGSI-LD API.

5.2.3.1 Basic Operations

The NGSI-LD API supports a number of operations, with messages expressed using JSON-LD.[NGS19] It allows context consumers and context producers to interact with context information systems. Not all conceivable operations are supported in the API, but rather a subset that is as simple as possible yet complex enough to handle most interactions.

For representation in the NGSI-LD API, any Entity is represented by a JSON-LD encoded object. The JSON-LD representation of an Entity includes a reference (in the @context statement) to the NGSI-LD Meta-Model (see Figure 9) along with the specific Entity Type Name, the Entity URI, the Properties and the Relationships associated with that Entity. Each Property includes a Property type, and a value (JSON data type or JSON object). Each Relationship includes a Relationship type, and the object of the Relationship (e.g., another entity). An important characteristic of NGSI-LD is that Properties and Relationships (which are together termed Attributes) may themselves also have Attributes.

Particularly, the API operations allow applications to discover, query and explore the graph-based data by specifying any combination of entities, types, relationships and/or properties as criteria for data queries.

An HTTP REST binding of the NGSI-LD operations is defined in the NGSI-LD API specification. [NGS1]

5.2.3.2 Context Producers and Consumers

One group of NGSI-LD operations allows Context Producers to create NGSI-LD Entities, i.e. insert an object with a defined URI into the system, and also allows Context Consumers to retrieve and subscribe to Entities.

These are as follows:

- Context Information Provision – a set of operations through which a Context Producer can create, modify, and delete an NGSI-LD Entity.
- Context Information Consumption – operations through which a Context Consumer can retrieve or query for NGSI-LD Entities. Queries can filter out Entities by Attribute Values (target value of a Property or the target value of a Relationship).
- Context Information Subscription – operations through which regular or event-driven update notifications of the context of one or more Entities can be created, updated, retrieved, queried for.

The following diagram (Figure 9) exemplifies these operations:

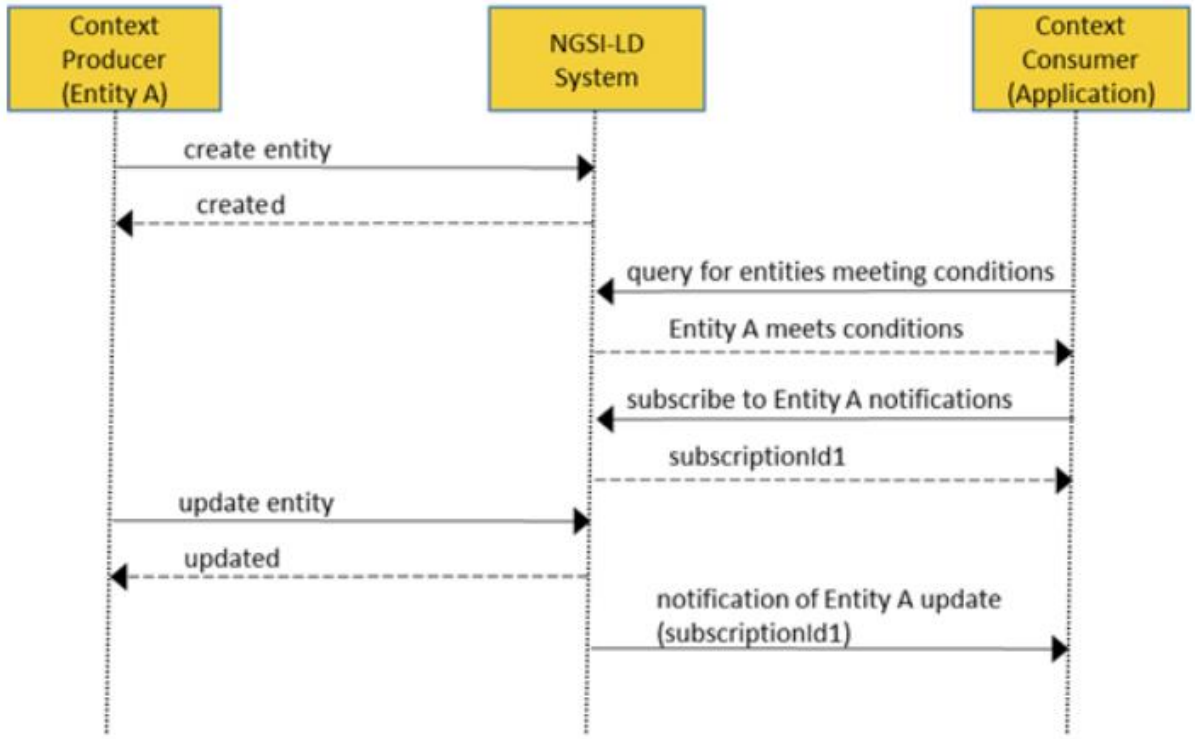


Figure 9. Example API Operations (Context Producer and Context Consumer)

5.2.3.3 Context Sources and Consumers

Another group of NGSI-LD operations allows Context Sources to be registered as potential sources of information meeting certain conditions. A Distribution Broker can query a Registry to ascertain which Context Sources may

be able to provide the information requested.

- Context Source Registration – a set of operations through which a Context Source (i.e., the entire collection of information which it could provide) can be registered, updated, and deleted (removed from the registry). The registration information includes the types of Entities, Properties, and Relationships about which the Context Source can provide information, as well as geographic and temporal constraints on the information (e.g., “only in the region Germany”, “only for years 2017 and later”). For example, a Context Source could register that it can provide the indoor temperature for Building A and Building B or that it can provide the speed of cars in a geographic region covering the centre of a particular city.
- Context Source Discovery – operations through which a Context Consumer or Producer can retrieve or query Context Source registrations.

An example of a context source and consumer is presented in Figure 10:

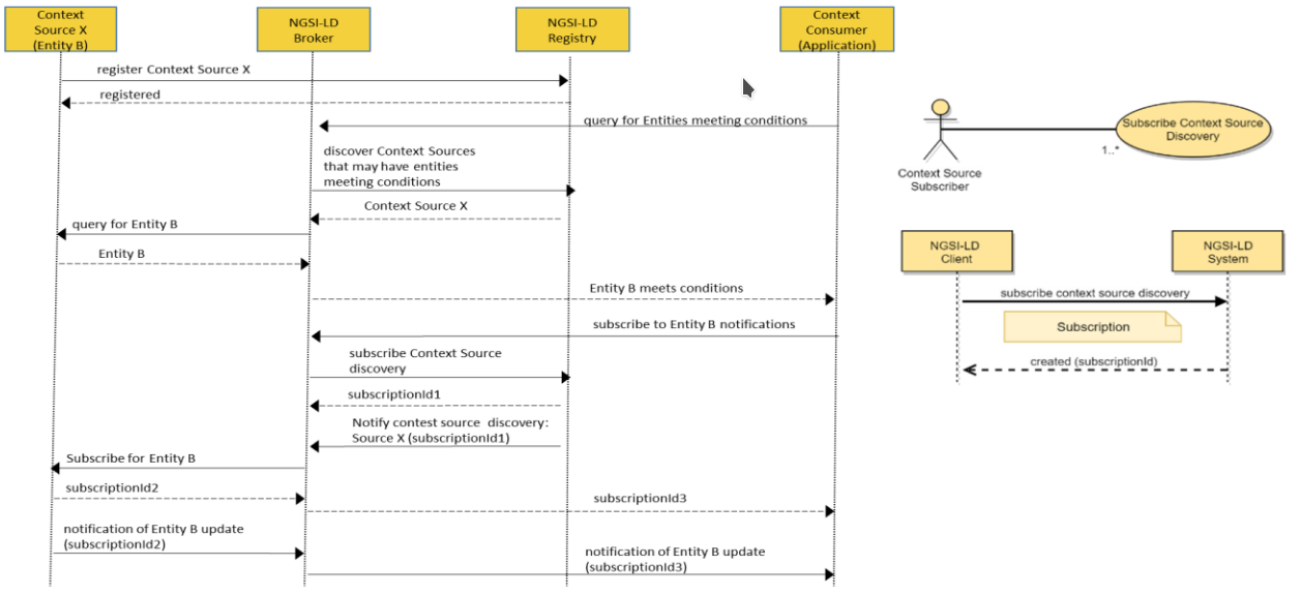


Figure 10. Sample API Operations (Context Source and Context Consumer)

5.3 FIWARE AgriFood Data Model

Having presented NGSI and NGSI-LD in the previous sections, we can now present the data models used in the domain of Smart Agrifood.

The FIWARE data models have been harmonized to enable data portability for different applications, especially in the domains of Smart Cities and Smart Agrifood. They are intended to be used together with FIWARE NGSI version 2 and NGSI-LD.

The FIWARE Agrifood data model represents a standard format which provides support to develop FIWARE solutions[5] in the domain of Smart Agrifood. The adoption of a standard model improves significantly the standardization of information coming from IoT Networks (including IoT sensors, wearables, GPS services, UAVs, robots and drones) and farm machines and, at the same time, it increases the uniformity and interoperability of the data within the FIWARE technologies and applications ecosystem.

This model can cover and map a whole series of information (among the most significant) coming from the IoT Sensor Network: information relating to animal welfare observation, crop pest and disease management, observations regarding weather forecasts, etc.

Like any other FIWARE Data Model[6], the specific one for the Smart Agrifood was implemented by following a series of guidelines[7] that outline specifications that every FIWARE DATA Model must follow to be compliant to the standard (i.e. the syntax, the clauses for reuse, definitions related to the NGSI protocol and consequently to the context information). Furthermore, these directives describe the physical structure of the model, the compliance with the specifications of the NGSI and NGSI-LD protocols (FIWARE data models, in most cases, are compliant with the NGSI v2 coding) and consequently the particular restrictions in the use of keywords (as in the case of attributes or types of data) or in the use of specific standards. This model will also follow the evolution of the FIWARE platform, while the NGSI-LD mappings will be added to all existing medium-term data models.

Each Data Model is programmatically defined using a JSON Schema. The JSON Schema covers only the so-called *key-value* representation of NGSI v2 context data. Thus, the JSON Schema does not cover the *normalised* representation of context data.

To switch from one format to another, some FIWARE scripts are available in Phyton (https://github.com/FIWARE/data-models/tree/master/tools).

Official validators are also available. The FIWARE Data Model validator is an utility to assist in the management of NGSI Data Models (https://github.com/FIWARE/data-models/tree/master/validator).

The FIWARE Data Models used in the AgriFood sector are presented below. All of them are primarily associated with the agricultural vertical and related IoT applications:

- AgriApp
- AgriCrop
- AgriFarm
- AgriGreenhouse
- AgriParcel

[5] https://www.fiware.org/community/smart-agrifood/
[6] https://www.fiware.org/developers/data-models/
[7] https://github.com/FIWARE/data-models/blob/master/specs/guidelines.md

- AgriParcelOperation
- AgriParcelRecord
- AgriPest
- AgriProductType
- AgriSoil
- Animal
- WeatherObserved
- WeatherForecast

5.3.1 AgriApp

This entity contains a harmonised description of a generic app made for the Agrifood domain.

The Data Model has the following properties:[8]

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriApp” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the application,
- description: Description of application,
- version: Version of the application,
- hasProvider: Provider (Person or Organization) of the application,
- endpoint: Endpoint URL of the application,
- category: ["TBD"][9]

5.3.2 AgriCrop

This entity contains a harmonised description of a generic crop.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriCrop” (required)

[8] For each property given here (and in the next subsections), either the values are given as specific examples, or there is an explanation of the type of object or value that could be given for each property.
[9] FIWARE is work in progress, so in some properties it has values To Be Determined.

- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the crop (required)
- alternateName: Alternative name (Scientific name) of the crop,
- agroVocConcept: URL of the FAO details of the crop,
- seeAlso: Other useful URL,
- description: General description of the crop,
- relatedSource: Reference application,
- hasAgriSoil: Relevant AgriSoil object
- hasAgriFertiliser: Relevant AgriFertilizer object(s)
- hasAgriPest: Relevant AgriPest object
- plantingFrom: Date Range of the planting,
- harvestingInterval: Date Range of the harvesting,
- wateringFrequency: Frequency of watering (choosing from: "daily", "weekly", "biweekly", "monthly", "onDemand", "other")

5.3.3 AgriFarm

This entity contains a harmonised description of a generic farm made up of buildings and parcels.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriFarm” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the farm,
- seeAlso: Other useful URL,
- description: General description of the farm,
- relatedSource: Reference application,
- location: Point of GPS coordinates of the farm,
- landLocation: Geometry defining the boundaries of the farmland,
- address: address of the farm,
- contactPoint: Contact information of the farm, i.e email, telephone, etc.,
- ownedBy: Owner (Person or Organization) of the farm,
- hasAgriParcel: Relevant AgriParcel object

5.3.4 AgriGreenhouse

This entity contains a harmonised description of the conditions recorded within a generic greenhouse, a type of AgriParcel.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriGreenhouse” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- ownedBy: Owner (Person or Organization),
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- belongsTo: Holder (Person or Organization),
- hasAgriParcelParent: Relevant AgriParcel object
- hasAgriParcelChildren: Relevant AgriParcel object
- hasWeatherObserved: Relevant Weather object
- hasWaterQualityObserved: Relevant WaterQuality object
- relativeHumidity: Humidity,
- leafTemperature: Temperature of the leaf,
- co2: The measured interior C02 concentration nominally in mg/L,
- dailyLight: Hours of light daily,
- drainFlow: Value, minimum, maximum and unit of measurement of rainfall,
- hasDevice: List of connected devices

5.3.5 AgriParcel

This entity contains a harmonised description of a generic parcel of land.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriParcel” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- location: Point of GPS coordinates of the parcel (required)

- area: Area of the parcel (required)
- category: Category of the parcel (ie. Arable),
- ownedBy: Owner (Person or Organization),
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- belongsTo: Holder (Person or Organization),
- hasAgriParcelParent: Relevant AgriParcel object
- hasAgriParcelChildren: Relevant AgriParcel object
- hasAgriCrop: required, Relevant AgriCrop object
- cropStatus: Status of the crop ("seeded", "justBorn", "growing", "maturing", "readyForHarvesting"),
- lastPlantedAt: Date of the last planted crop,
- hasAgriSoil: Relevant AgriSoil object
- hasDevice: List of connected devices

5.3.6 AgriParcelOperation

This entity contains a harmonised description of a generic operation performed on a parcel of land.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriParcelOperation” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- hasAgriParcel: required,
- description: General description of the operation,
- operationType: Type of operation ("fertiliser", "inspection", "pesticide", "water", "other")
- result: Result of the operation ("ok", "aborted", "failed"),
- plannedStartAt: Planned start date (required)
- plannedEndAt: Planned end date (required)
- status: Status of the operation ("planned", "ongoing", "finished", "scheduled", "cancelled"),
- hasOperator: Relevant Person object
- startedAt: Start date (and time),
- endedAt: End date,
- reportedAt: Reported date,

- hasAgriProductType: Relevant AgriProduct object
- quantity: Quantity associate to the operation,
- waterSource: Source of the water ("borehole", "rainfall", "river", "rainwater capture", "water dam", "commercial supply"),
- workOrder:
- workRecord:
- irrigationRecord:

5.3.7 AgriParcelRecord

This entity contains a harmonised description of the conditions recorded on a parcel of land.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriParcelRecord” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- hasAgriParcel: required
- location: Geometry of GPS coordinates of the record (required)
- description: General description of the record,
- soilTemperature: Temperature of the soil,
- soilMoistureVwc: Volumetric Water Content of the soil,
- soilMoistureEc: Electrical Conductivity of the soil,
- soilSalinity: Salinity of the soil,
- leafWetness: Wetness of the leaf,
- leafRelativeHumidity: Humidity to the leaf,
- leafTemperature: Temperature of the leaf,
- airTemperature: Temperature of the air,
- solarRadiation: Solar radiation,
- relativeHumidity: Humidity,
- atmosphericPressure: Pressure of the atmosphere,
- hasDevice: List of connected devices,
- observedAt: Observed date,
- depth: Metadata to indicate the associated depth where soil measurements are taken

5.3.8 AgriPest

This entity contains a harmonised description of an agricultural pest.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriPest” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the pest (required)
- alternateName: Alternative name (Scientific name) of the pest,
- agroVocConcept: URL of the FAO details of the pest,
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- hasAgriProductType: Relevant AgriProduct object (that the pest can infect)
- description: Description of the pest

5.3.9 AgriProductType

This entity contains a harmonised description of a generic agricultural product type. The AgriProductType includes a hierarchical structure that allows product types to be grouped in a flexible way.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriProductType” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the product type (required)
- description: Description of the product type,
- agroVocConcept: URL of the FAO details of the product type,
- category: category of the product type ("fertiliser", "cropNutrition", "cropProtection", "cropVariety", "harvestCommodity"),
- hasAgriProductTypeParent: Parent type,
- hasAgriProductTypeChildren: List of child type,
- root: Boolean that indicate if this is a parent (required)

5.3.10 AgriSoil

This entity contains a harmonised description of a generic soil.

The Data Model has the following properties:

- id: unique identifier (required)
- type: NGSI Entity type. It must be equal to “AgriSoil” (required)
- dateCreated: "2017-01-01T01:20:00Z",
- dateModified: "2017-05-04T12:30:00Z",
- name: Name of the soil (required)
- alternateName: Alternative name (Scientific name) of the pest,
- agroVocConcept: URL of the FAO details of the soil,
- seeAlso: Other useful URL,
- relatedSource: Reference application,
- hasAgriProductType: List of product type,
- description: Description of the soil

5.3.11 Animal

This entity contains a harmonised description of a generic animal.

The model for the animal entity has the following properties:

- id: unique identifier
- type: Entity type. It must be equal to “Animal”
- species: Species to which the animal belongs (“dairy cattle”, “beef cattle”, “sheep”, “goat”, “horse”, “pig”) (required)
- relatedSource: ID of the animal in external applications
- legalID: Legal ID of the animal (required)
- birthdate: Animal’s birthdate (required)
- sex: Sex of the animal: (“female”, “male”) (required)
- breed: Breed of the animal
- calvedBy: Mother of the animal
- siredBy: Father of the animal
- location: Location of the animal represented by a GeoJSON geometry.
- weight: The weight of the animal as a number

- ownedBy: The owner of the animal
- locatedAt: AgriParcel relationship
- phenologicalCondition: Phenological condition of the animal
- reproductiveCondition: Reproductive condition of the animal
- healthCondition: Health condition of the animal
- fedWith: Food used for the animal
- welfareCondition: Indicator of the animal welfare

5.4 GS1 standards and data model

Before we present any specific data and ontologies of data (in later section) which typically are quite heterogeneous, the fact that there are different data models for different data types (or similar data gathered from different sources and devices) makes apparent the need for a Global Data Model. This is the aim of GS1.

GS1 identification standards provide the means to identify real-world entities so that they may be the subject of electronic information that is stored and/or communicated by end users. The GS1 identification standards include unique identifiers (called GS1 identification keys), which may be used by an information system to refer unambiguously to a real-world entity such as a trade item, logistics unit, physical location, document, service relationship or any other entity.

GS1 standards for data capture provide the means to automatically capture data that is carried directly on physical objects, bridging the world of physical things and the world of electronic information. The GS1 data capture standards include:

- Definitions of barcode and radio-frequency identification (RFID) data carriers, which allow GS1 identification keys and supplementary data to be affixed directly to a physical object.
- Standards that specify consistent interfaces to readers, printers, and other hardware and software components that connect the data carriers to business applications. GS1 standards for data exchange provide the means to share information, both between trading partners and internally, providing the foundation for electronic business transactions, electronic visibility of the physical and digital world, and other information applications.

GS1 standards for information sharing are:

- Definitions of master data, business transaction data and physical event data.
- Tools for optimising online product search
- Communication standards for sharing this data between applications and trading partners
- Discovery standards that help locate where relevant data resides across a supply chain
- Trust standards that help establish the conditions for sharing data in a secure way.

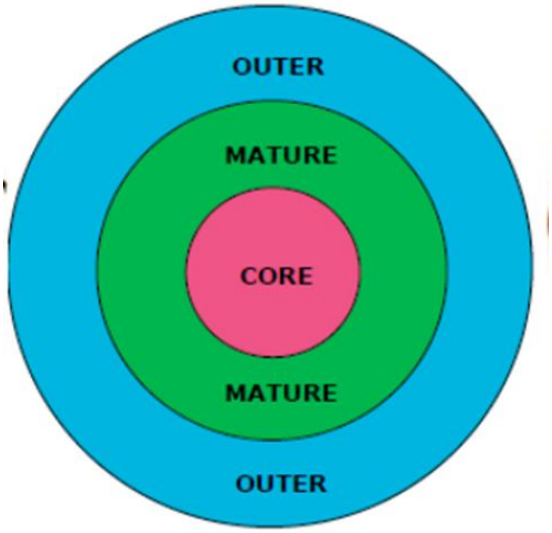


Figure 11. The "Onion Model" paradigm

GS1 in Europe has agreed with users acting at the European level, to start working on regional data models using "The Onion Model" (Figure 11) since 2016 with the full support of the European associations (AIM, Eurocommerce, Food & Drink Europe, wines and spirits). On top, important work has been happening in North America (SmartLabel) and in other regions and sectors. There is now an opportunity to bring all this work together at the global level.

When you cut an onion, you will find different layers. And this is how the global data model is built: it consists of a core (heart), mature and outer layer. Data attributes are divided into 3 groups visualized like an onion:

- Core layer attributes: attributes that are used in all analysed markets, regardless of the product category;
- Mature layer attributes: attributes that are used in most of the data models (depending on the product category) worldwide or at regional level;
- Outer layer attributes: attributes that are only optional and used in the national market for legal or national business reason.

5.5 Saref4agri

This section is a technical specification of Saref4Agri, which is an OWL-DL ontology that extends SAREF for the Smart Agriculture and Food Chain domain. STF 534[10], an ETSI specialist's task force that was established with the goal to extend the SAREF ontology for the domains of Smart Cities, Smart Industry & Manufacturing and Smart

[10] https://portal.etsi.org/STF/stfs/STFHomePages/STF534

AgriFood[11], develops this ontology. The intention of Saref4Agri, as mentioned in the associated Saref4Agri requirements document ETSI TR 103 511, is to connect SAREF with existing ontologies (such as W3C SSN, W3C SOSA, GeoSPARQL, etc.) and with important standardization initiatives and ontologies in the Smart Agriculture and Food Chain domain, including ICAR that is used for livestock data[12], AEF for agricultural equipment[13], Plant Ontology Consortium for plants[14], and AgGateway for IT support for arable farming[15].

To show the potential of Saref4Agri, this section focuses on two examples, which are the "livestock farming" and "smart irrigation" use cases. Various other examples exist in the Smart Agriculture and Food Chain domain, such as arable farming, greenhouses, horticulture, agricultural equipment and food chain. For an exhaustive list of use cases, see also the H2020 Large Scale Pilot "Internet of Food and Farm 2020 (IoF2020)" at https://iof2020.eu/trials. However, it was necessary to make actionable choices within the STF 534 timeframe and the available resources; thus, livestock farming and smart irrigation have been chosen as the two initial examples to create Saref4Agri. As a next step, it is recommended to further refine the proposed livestock farming and smart irrigation examples to add relevant sensors that are not considered yet, and also consider additional use cases to create new releases of Saref4Agri.

As all the SAREF ontologies, Saref4Agri is a dynamic semantic model that is meant to evolve over time. Therefore, the stakeholders in the AgriFood domain (starting from the ICAR, AEF and AgGateway initiatives) are invited to use, validate and provide feedback on Saref4Agri, collaborating with the SAREF ontology experts to improve and evolve Saref4Agri in an iterative and interactive manner, so that changes and additions can be incorporated in future releases of the present document.

The livestock farming and smart irrigation use cases, used as a basis to create Saref4Agri in the present section, are concerned with the integration of multiple data sources for the purpose of providing decision support services located on the local "Farm Management System" of the farmers or provided by a service over the network. Multiple data sources of interest include GPS, meteorological data (both historic and current), remote observation (via satellite sources such as Copernicus) and local observation using near or proximal sensors.

As an extension of SAREF, which is a semantic model for IoT that describes smart devices and applications in terms of their functions, services, states and measurements, Saref4Agri is concerned with the description of proximal sensors that measure a variety of relevant parameters for agriculture, including: (about animals) movement, temperature, etc., (about soil) moisture/humidity, Ph value, salinity, compaction, (about plants) plant colour (NDVI), etc. The measurements from these sensors need to be integrated by a decision support service to enable the planning of (for example) a treatment plan for animals (in a livestock scenario), or a decision

[11] https://portal.etsi.org/STF/stfs/STFHomePages/STF534
[12] https://www.icar.org/
[13] http://www.aef-online.org
[14] http://archive.plantontology.org
[15] http://www.aggateway.org/

to irrigate or harvest (in an irrigation, horticulture or greenhouse context). The requirements used to create the Saref4Agri extension specified in the present document are described in the associated ETSI TR 103 511.

An overview of the Saref4Agri ontology is provided in Figure 12. If the element is defined in Saref4Agri, the prefix is s4agri. Arrows are used to represent properties between classes and to represent some RDF, RDF-S and OWL constructs, more precisely:

- Plain arrows with white triangles represent the rdfs:subClassOf relation between two classes. The origin of the arrow is the class to be declared as a subclass of the class at the destination of the arrow.
- Dashed arrows between two classes indicate a local restriction in the origin class, i.e. that the object property can be instantiated between the classes in the origin and the destination of the arrow. The identifier of the object property is indicated within the arrow.
- Dashed arrows with identifiers between stereotype signs (i.e. "<< >>") refer to OWL constructs that are applied to some ontology elements, that is, they can be applied to classes or properties depending on the OWL construct being used.
- Dashed arrows with no identifier are used to represent the rdf:type relation, indicating that the element in the origin of the arrow is an instance of the class in the destination of the arrow.
- Datatype properties are denoted by rectangles attached to the classes, in a UML-oriented way.
- Dashed boxes represent local restrictions in the class, i.e. datatype properties that can be applied to the class they are attached to.
- Individuals are denoted by rectangles in which the identifier is underlined.

Note that Figure 12 aims at showing a global overview of the main classes of Saref4Agri and their mutual relations.

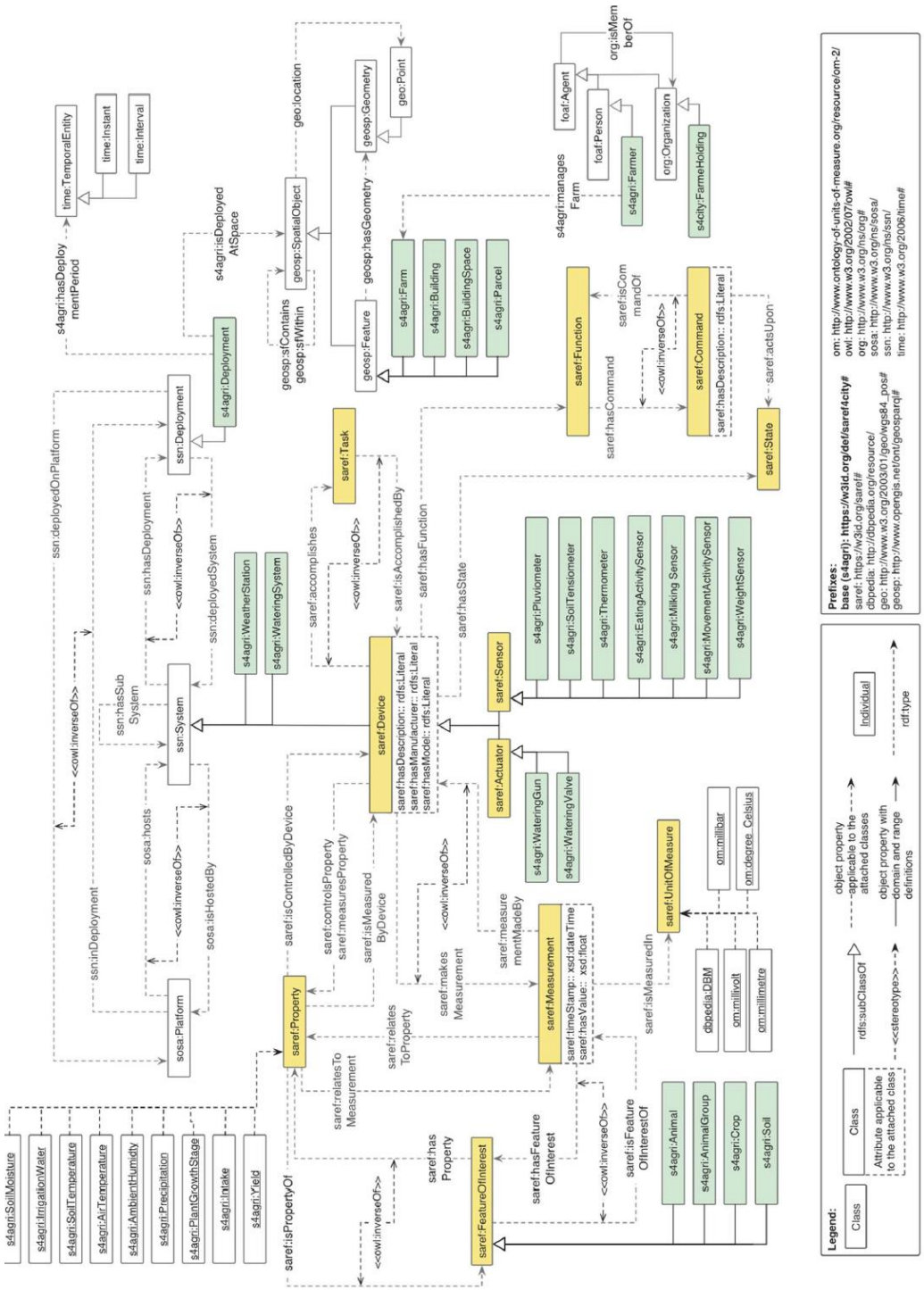


Figure 12. Saref4Agri ontology

5.6 INSPIRE data model for Agricultural and Aquaculture Facilities

The Agricultural and Aquaculture Facilities (hereinafter AF) [AF13] model is composed of core information in relation to the geographical description of entities under the Agriculture and Aquaculture scope. Agricultural and aquaculture facilities are in the INSPIRE Directive defined as "farming equipment and production facilities (including irrigation systems, greenhouses and stables)".

The AF data model is based on the Activity Complex model [AC13]. “Activity Complex” is in INSPIRE a generic name agreed across thematic domains trying to avoid specific thematic connotations such as “Plant”, “Installation”, “Facility”, “Establishment” or “Holding”. Because of this, the Activity Complex model must adhere to the requirements of horizontal datasets, in which facilities are considered independently of their thematic scope. Such scope may be for the DEMETER project the Nitrate Directive, Water Framework Directive or Waste Directive. An overview of the feature types and data types of AF data model are the following:

- Activity Complex: The whole area and all infrastructures on it under the control of an operator. In the AF theme, the Activity Complex has the specialised representation named Holding.
- Holding: The whole area and all infrastructures included on it, under the control of an operator to perform agricultural or aquaculture activities. It may be composed of one or more Sites.
- Site: Belonging to a holding, it is the geographical representation of land that constitutes a management unit. It includes all infrastructure, equipment and materials.
- All "Holding" must be related to at least one "Site" but a Holding can manage one or more "Sites".
- The geographical extension of the "Site" has been described as GM_Object to allow its representation as a point (inherited from holding) or more complex representations as a Set of Isolated Polygons (Multi-surface). Topologically all the rest of geographical elements should be included under the limits of one "Site". If necessary, a "Site" should be created to include each of these representative sub-elements.
- The location of “Holding” and “Site” can be expressed as a point or a surface (polygon).
- The activity of “Holding” and “Site” is expressed by using the standard NACE (NAtional Classification of Economic activities) classification list[16].
- The data type FarmAnimalSpecies is expressed by using a standard code list for livestock animal species and a standard FAO code list for aquaculture species.

[16] http://ec.europa.eu/competition/mergers/cases/index/nace_all.html

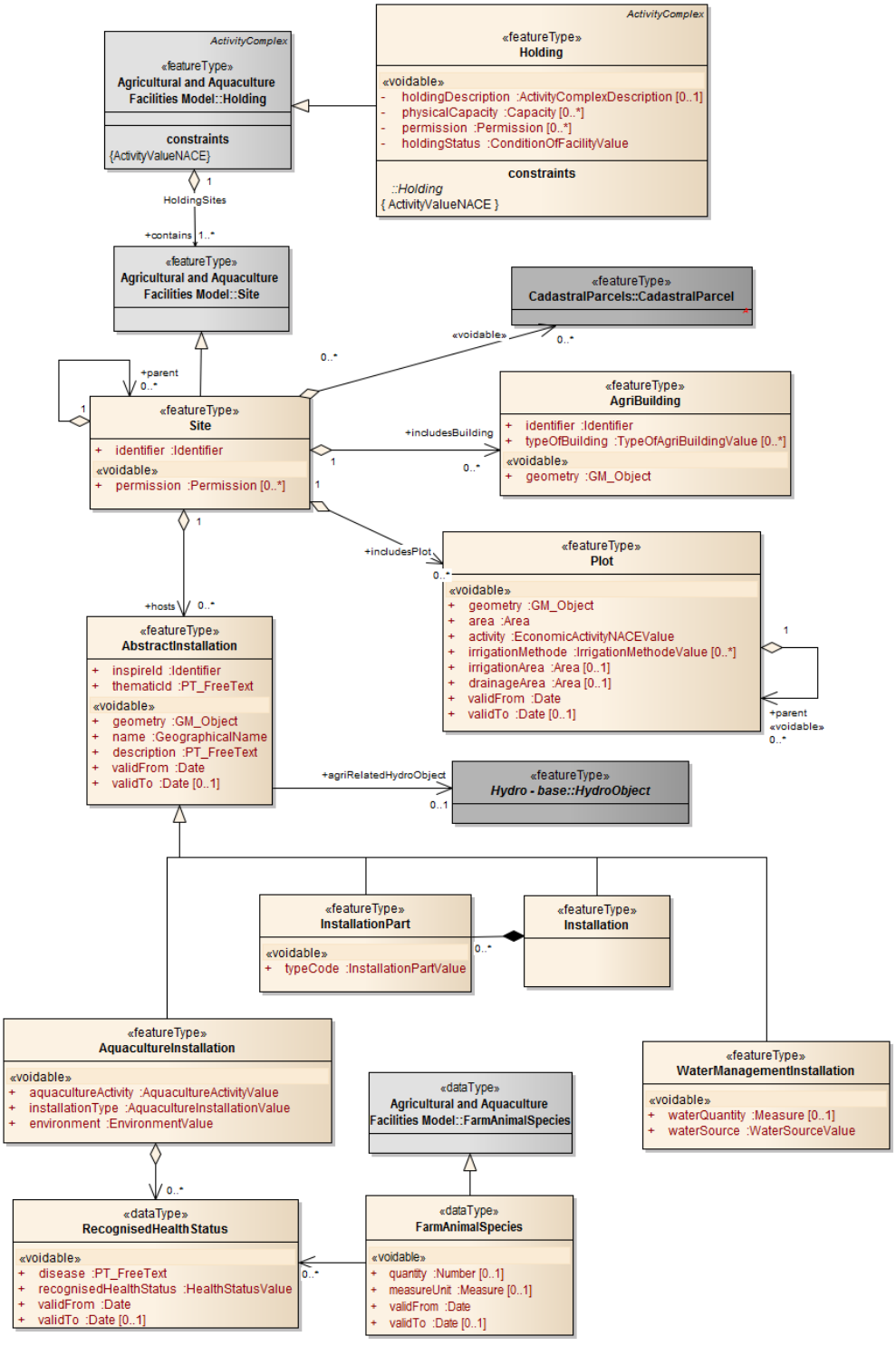


Figure 13. Overview of the AF Extended Model, feature classes

The AF specification also includes an extended model to represent complementary information about Agricultural and Aquaculture Facilities. The latest specification release (end 2013) includes extensions about plots, agri-buildings, installations, irrigation and drainage, farm animals and animal health. An overview of the feature classes of the extended model is depicted in the Figure above and is described below:

- Installation: refers to all technical instruments and constructions included on the "Site" that should be described independently. It allows referring to specific sub-elements included on the "Site" and legally related to the "Holding".
- Plot: allows describing in an abstract sense delimited portions of land or water (independently of their size or delimitation method) included on a "Site" dedicated to a specific function as part of a major activity and geographically identifiable. "Plot" concept shouldn't be confused with Cadastral entities, despite the fact that, in some cases, it could be coincident on the real world with them.
- AgriBuilding: The relation between "Buildings" and specific uses is quite fuzzy and, for this reason, only buildings dedicated to specific functions related to the Activity should be linked with the AF model, otherwise the consistency of datasets could be quite complicated.
- HydroObject: The relation between "HydroObject" and "Installation" illustrates the link between the hydrographic system (irrigation and drainage systems) of the "Site" and the natural hydro objects, like ponds, lakes, rivers and canals, which are identified by theme Hydrography.

5.7 The rmAgro model

The model rmAgro has several packages:

- A Business Process Model, which specifies some processes from the European FiSpace project.
- An ontology, the OWL model as exercise for an OWL version of the domain model.
- A use Case model, with some use cases from ISO/TC23/SC19/WG5 on wireless communication around fleet management.
- The domain model, which is the core of the reference model.
- A dynamic view with some sequence diagrams from FiSpace DDL model.
- The database model as result of transformation from the domain model.
- The deployment model, used to specify some platform classes used in agriculture.
- External models, with specification of some third-party models which could be imported in EA.
- External XSD's, with some third party xsd specifications, which could be imported from xsd file in EA Java model Agro with a java interface model and a java implementation model as result of transformation from the domain model.
- Mapping, with diagrams in which some mapping of third-party models are visualized.
- WSDL's, which specifies the messages used in the FiSpace project.
- XSD model Agro, as result of transformation from the domain model.

- drmAgro, a generic part of the domain model, which holds classes that are applicable for all branches of agriculture. It has separate packages for cropping, animal husbandry, greenhouse production, post-harvest processing and infrastructure. Apart from this branch specific subpackages, there are subpackages for data types, enumerations, computer platforms, geometries, xsd types and swe (sensor web enablement) types. In drmAgro and its subpackages there are diagrams for different scopes of the model. For each class and attribute definitions are given, with eventual additional remarks and examples. The result is a public available reference model which is and will be in continuous development. In early 2017 drmAgro held 38 generic classes and drmCrop 239 specific classes, this last one containing quite a number of classes which could be moved to the generic part. Crop production covers a wide range of use cases like planning and reporting of fieldwork, advices, soil sampling and analyses, application of crop growth models, scheduling of farm operations, auditing, etc.

This reference model is the basis for standardised XML messages exchanged between farm management systems and advisors, processors and the government in the Netherlands. It is also the basis for data exchange through the FIspace platform developed as an EU project.

5.8 Semantic Sensor Network (SSN) ontology

In this section, we present one more ontology related to the agrifood domain. This section presents an ontology related to sensors and their measurements, and more specifically the Sensor, Observation, Sample, and Actuator (SOSA) Semantic Sensor Network (SSN) ontology.

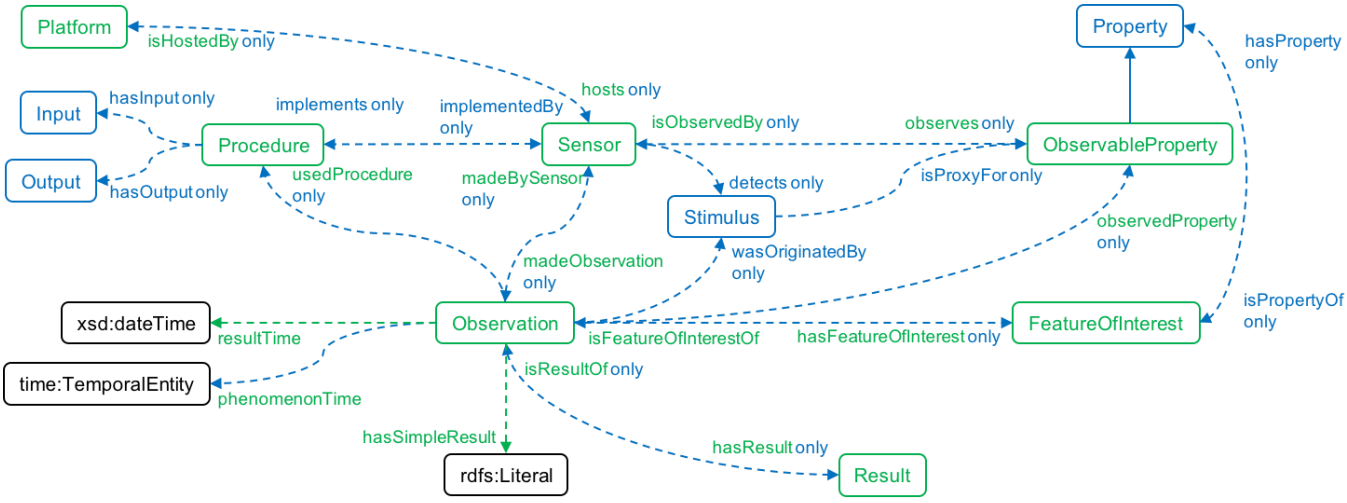


Figure 14. Semantic Sensor Network: Observation Model[17]

[17] source: https://www.w3.org/TR/vocab-ssn/images/SSN-Observation.png

The Semantic Sensor Network (SSN) Ontology[18] has been jointly developed by the Open Geospatial Consortium together with W3C as part of the Spatial Data on the Web Working Group (SDWWG)[19] - a previous joint W3C/OGC project now succeeded by the Spatial Data on the Web Interest Group[20]. OGC is currently seeking comment as to whether the specification should also be approved as OGC standard.

The Semantic Sensor Network (SSN) ontology is used for describing actuators, sensors and their observations, the involved procedures, the studied features of interest, the samples used to do so, and the observed properties. SSN follows a horizontal and vertical modularization architecture by including a lightweight but self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties. With their different scope and different degrees of axiomatization, SSN and SOSA can support a wide range of applications and use cases, including satellite imagery, large-scale scientific monitoring, industrial and household infrastructures, social sensing, citizen science, observation-driven ontology engineering, and the Web of Things.

Figure 14 above provides an overview of the core classes and properties that are specifically related to modelling observations. SOSA axioms are shown in green, while SSN-only axioms are shown in blue. All classes are described in full detail in the W3C recommendation.

5.8.1 SensorThings

The OGC SensorThings API[21] is an OGC standard specification for providing an open and unified way to interconnect Internet of Things (IoT) devices, data, and applications over the Web. It is an open standard, builds on Web protocols and the OGC Sensor Web Enablement standards, and applies an easy-to-use REST-like style. As a result, it provides a uniform way to expose the full potential of the IoT.

At a high level, the OGC SensorThings API provides two main functionalities and each function is handled by specific parts: the two parts are the Sensing part and the Tasking part. The Sensing part provides a standard way to manage and retrieve observations and metadata from heterogeneous IoT sensor systems. The Tasking part is planned as a future work activity.

The SensorThings API's data model is based on OGC/ISO Observations and Measurements (OGC/ISO 19156:2011), so that it can easily interoperate with OGC Sensor Observation Services (SOS). Also note that SensorThings is RESTful, uses JSON encoding, adopts the OASIS OData URL pattern and query options, and supports the ISO MQTT messaging protocol.

[18] https://www.w3.org/TR/vocab-ssn/
[19] http://www.opengeospatial.org/projects/groups/sdwwg
[20] https://www.w3.org/2017/sdwig/
[21] http://www.opengeospatial.org/standards/sensorthings

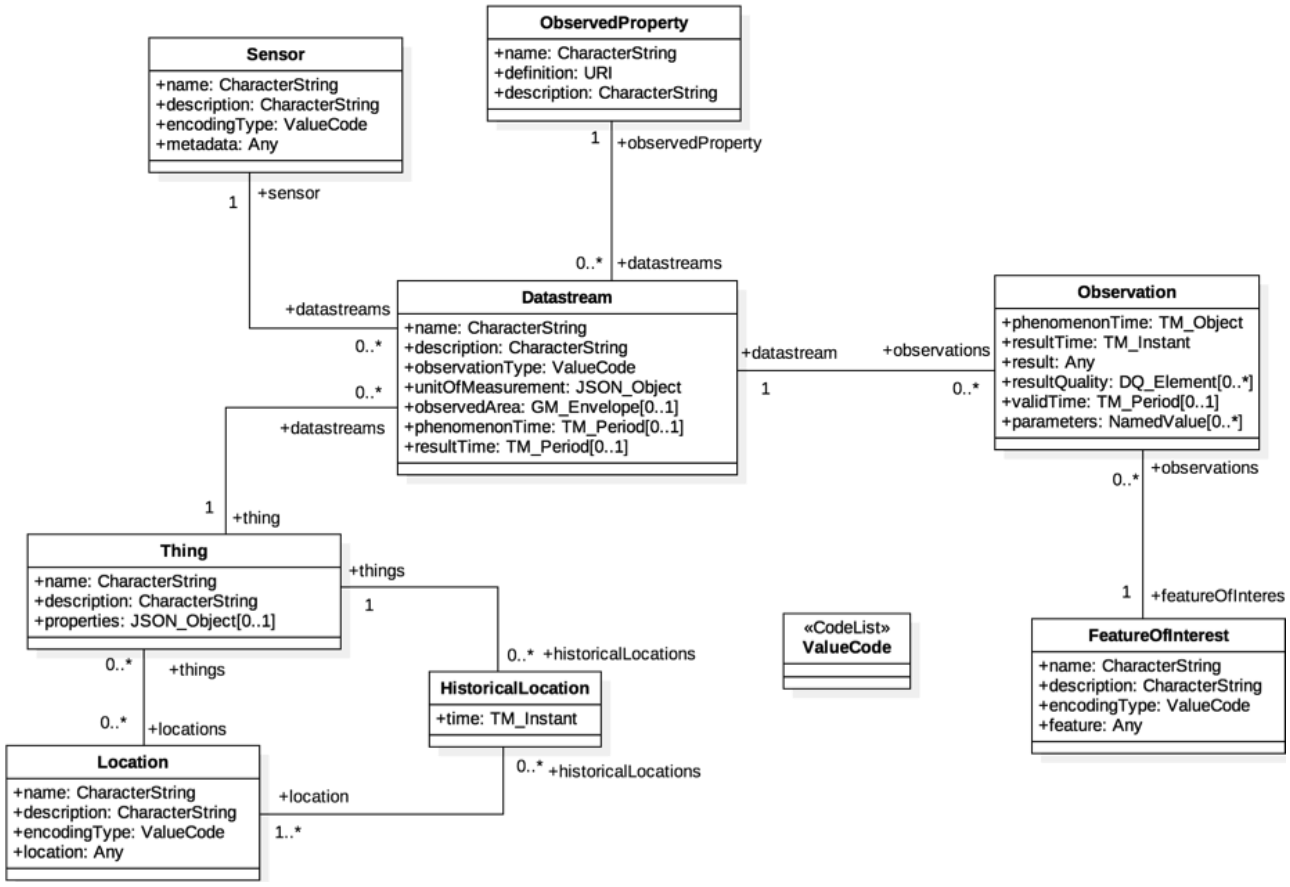


Figure 15. SensorThings Datastream[22]

SensorThings API defines eight entities (Figure 15) for the IoT sensing applications. These entities are Thing, Datastream, Sensor, Observation, ObservedProperty, FeatureOfInterest, Location and HistoricalLocation. The figure illustrates these putting focus on the central Datastream entity.

5.8.2 Observation and Measurements

The Observation and Measurements (O&M) conceptual model is an OGC Abstract Specification [OnM11] and identical with ISO 19156:2011 Geographic information -- Observations and measurements. OGC further specifies an XML implementation of the conceptual model including a schema for Sampling Features. This encoding is an

[22] source: http://docs.opengeospatial.org/is/15-078r6/15-078r6.html

essential dependency for the OGC Sensor Observation Service (SOS) Interface Standard. More specifically, this standard defines XML schemas for observations, and for features involved in sampling when making observations. These provide document models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities. Recently, work has started in OGC to develop a JSON encoding of O&M.

O&M defines a conceptual schema for observations, and for features involved in sampling when making observations. These provide models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities. Observations commonly involve sampling of an ultimate feature-of-interest. O&M defines a common set of sampling feature types classified primarily by topological dimension, as well as samples for ex-situ observations. The schema includes relationships between sampling features (sub- sampling, derived samples).

O&M concerns only externally visible interfaces and places no restriction on the underlying implementations other than what is needed to satisfy the interface specifications in the actual situation.

Figure 16 illustrates the O&M model.

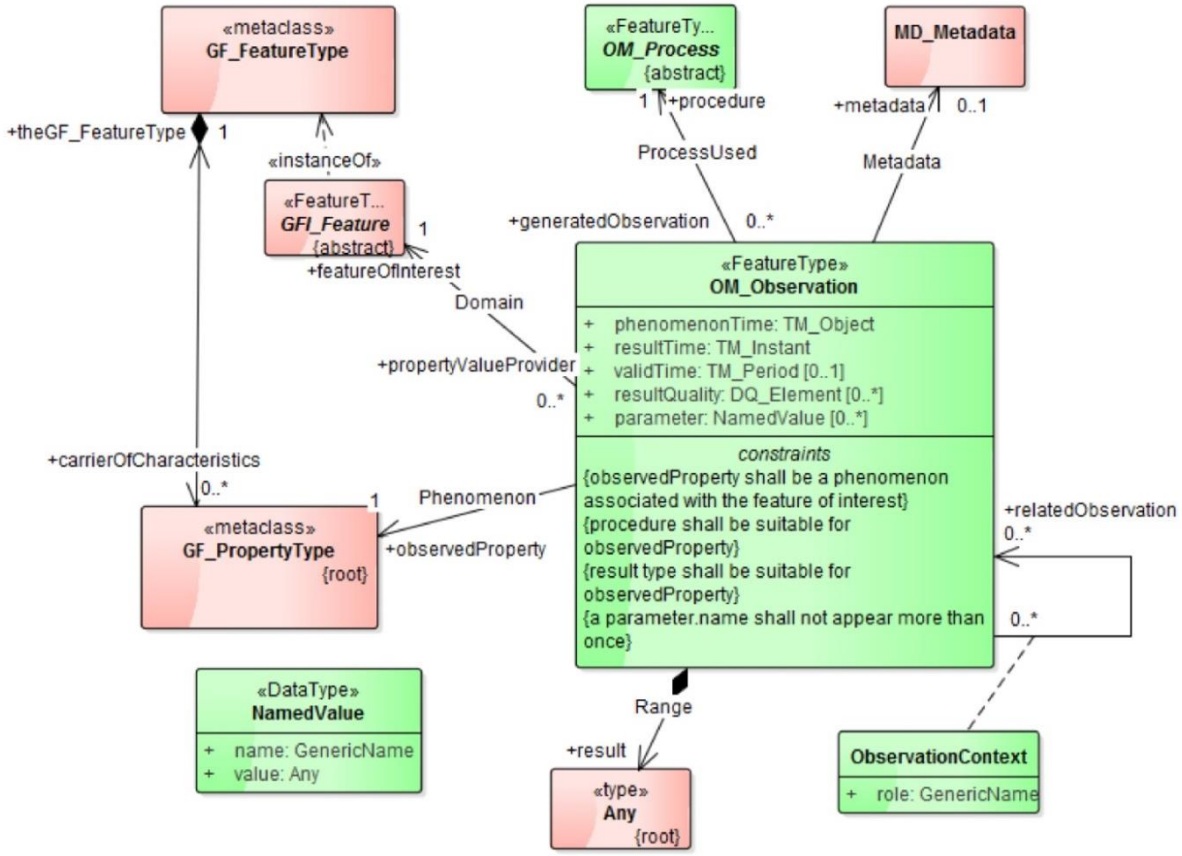


Figure 16. O&M Observation model[23]

5.8.3 Alignment of O&M, SSN/SOSA, and SensorThings

SSN/SOSA and SensorThings are both aligned with Observations and Measurements. However, the alignment uses different approaches and is not identical. There are a number of parts of SOSA/SSN relating to sampling and actuating, as well as procedures, platforms, etc., that are not part of SensorThings and await specific proposals as to how they might be incorporated.

As an example, SensorThings "Datastream" does not exist in SOSA/SSN, but may align with the SOSA:ObservationCollection proposed by the SSN extension[24]. SensorThings "Thing" does not exist in SOSA/SSN and it is unclear how that would be aligned. A potential solution could be to align SensorThings "Thing" with SOSA:Platform. This allows commonality of location for some number of Datastreams, Sensors, ObservableProperties, etc., and is relatively agnostic to whether it moves between observations as well as whether observations made from it target the same or different features of interest. Only slightly complicated by the fact that there is no direct connection as in SOSA between Sensor and Thing, only through Datastream. Sosa:Platform could be added as a common property to sosa:ObservationCollection to strengthen this alignment.

Another alignment issue is that not all SOSA relations go through Observation and the proposed SOSA extension sosa:ObservationCollection. A sos:Sensor observes a sosa:ObservableProperty. A sosa:ObservableProperty isPropertyOf sosa:FeatureOfInterest (through ssn:Property). sosa:Platform (isSameAs stapi:Thing ?) hosts sosa:Sensor. Granted that the relation chain through stapi:Datastream can be followed in STAPI, but it does require the existence of an appropriate Datastream in order to do so.

The CYBELE project looks further into the alignment between SSN/SOSA, SensorThings, O&M, SensorML (a model and encodings to describe sensors). It is of course important to integrate the different approaches commonly used and to provide a solid foundation for future adaptation.

5.9 AGROVOC

AGROVOC[25] is a controlled vocabulary covering all areas of interest of the Food and Agriculture Organization (FAO)[26] of the United Nations, including food, nutrition, agriculture, forestry, fisheries, scientific and common names of animals and plants, environment, biological notions, techniques of plant cultivation and more. It is published by FAO and edited by a community of experts. It consists of 36,000+ concepts available in up to 33

[23] source: OGC Topic 20: Observations and Measurements
[24] https://w3c.github.io/sdw/proposals/ssn-extensions/
[25] http://aims.fao.org/vest-registry/vocabularies/agrovoc
[26] http://www.fao.org/home/en/

languages. AGROVOC is both an RDF/SKOS-XL concept scheme, and a Linked Open Data (LOD) set, as described below.

AGROVOC is widely used in specialized libraries as well as digital libraries and repositories to index content and for the purpose of text mining. It is also used as a specialized tagging resource for knowledge and content organization by FAO and other third-party stakeholders.

All concepts of the AGROVOC thesaurus are hierarchically organized under 25 top concepts. AGROVOC top concepts are very general and high-level concepts, like “activities”, “organisms”, “location”, “products”, etc. More than half of the total number of concepts (20,000+) fall under the top concept “organism”, which confirms how AGROVOC is largely oriented towards the agricultural sector.

As an RDF/SKOS-XL concept scheme, the conceptual and terminological level are separated. The basic notions for such a concept scheme are concepts, their labels and their relations:

- Concepts: anything we want to represent or “talk about” in our domain. Concepts are represented by terms. A concept could also be considered as the set of all terms used to express it in various languages. In SKOS, concepts are formalized as skos:Concept, identified by dereferenceable URIs. For example, the AGROVOC concept with URI http://aims.fao.org/aos/agrovoc/c_12332 is for maize.
- Terms: the actual terms used to name a concept. For example, maize, maïs, 玉米 and मक्का are all terms used to refer to the same concept in English, French, Chinese and Hindi respectively. Terms are expressed by means of the SKOS extension for labels, SKOS-XL. The predicates used are: skosxl:prefLabel, used for preferred terms (“descriptors” in thesaurus terminology), and skosxl:altLabel, used for non- preferred terms.
- Relations: In SKOS, hierarchical relations between concepts are expressed by the predicates skos:broader, skos:narrower. They correspond to the classical thesaurus relations broader/narrower (BT/NT). Non-hierarchical relations express a notion of “relatedness” between concepts. AGROVOC uses the SKOS relation skos:related (corresponding to the classical thesaurus RT), and a specific vocabulary of relations called Agrontology[27] . AGROVOC also allows for relations between labels (i.e. terms), thanks to the SKOS-XL extension to SKOS.

AGROVOC is aligned with 18 other multilingual knowledge organization systems related to agriculture. The linked data version of AGROVOC is exposed as RDF and HTML, through a content-negotiation mechanism. It is also exposed through a SPARQL endpoint (Note: the original endpoint is at the moment of writing unavailable, but the dataset has been replicated in FOODIE repository[28]).

The advantage of having AGROVOC published as Linked Data is that once other thesauri are linked, the resources

[27] http://aims.fao.org/agrovoc/agrontology
[28] https://www.foodie-cloud.org/sparql

they index are linked as well. For instance, AGRIS provides a mash-up web application that links the AGRIS bibliographic repository (indexed with AGROVOC) to related web resources (indexed with vocabularies linked to AGROVOC).

5.10 FOODIE

FOODIE ontology [PaRe16] provides an application vocabulary covering different categories of information dealt by typical farm management tools/apps for their representation in semantic format, and in line with existing standards and best practices (INSPIRE, ISO/OGC standards). The model (result from FOODIE project) has been consulted and received positive feedback from experts in various institutions, including the EU DG JRC, EU Global Navigation Satellite Systems Agency (GSA), Czech Ministry of Agriculture, Global Earth Observation System of Systems (GEOSS), German Kuratorium für Technik und Bauwesen in der Landwirtschaft (KTBL).

The goal of the FOODIE ontology was to enable the representation of farm-related data in a semantic format, as well as to enable the exploitation of semantic technologies for different tasks. Some of these tasks may include: the transformation of (semi-)structured data to semantic format; ontology-based data access (e.g., accessing relational databases as virtual, read-only RDF graphs); interlinking data with established vocabularies (e.g., AGROVOC) and relevant datasets in the Linked Open Data (LOD) cloud; data integration using linked data as a federated layer; knowledge discovery through inference.

The ontology was generated from the FOODIE data model, an extension of the INSPIRE Agriculture and Aquaculture Facilities theme data model specification, which in turn is based on ISO/OGC standards for geospatial services and formats, thus applying the ISO/OGC-approach of modelling physical things, so-called “features”. Technically, INSPIRE specifications are defined as UML models and are available in different XML-based formats (e.g., GML, XMI) or as Enterprise Architect (EA) projects, and, in the same line, the FOODIE data model was specified in UML.

Hence, in order to build the ontology, it was necessary to transform or lift the FOODIE data model into a semantic format. This process was conducted semi-automatically by reusing existing tools and adhering to the mapping rules for transforming geographic information UML models to OWL ontologies defined by the ISO 19150-2 standard. In particular, we used the ShapeChange tool and addressed several issues and customizations before and after the execution of the ShapeChange processor as described in [PaRe16].

The FOODIE ontology is publicly available, and its top hierarchy is depicted in Figure 17. Similarly, the ontology classes along with their top parent stereotype class are depicted in Figure 18, Figure 19 and Figure 20, for FeatureType, Datatype and Codelist classes, respectively. Finally, Figure 21 depicts a partial view of the ontology core classes and their relations.

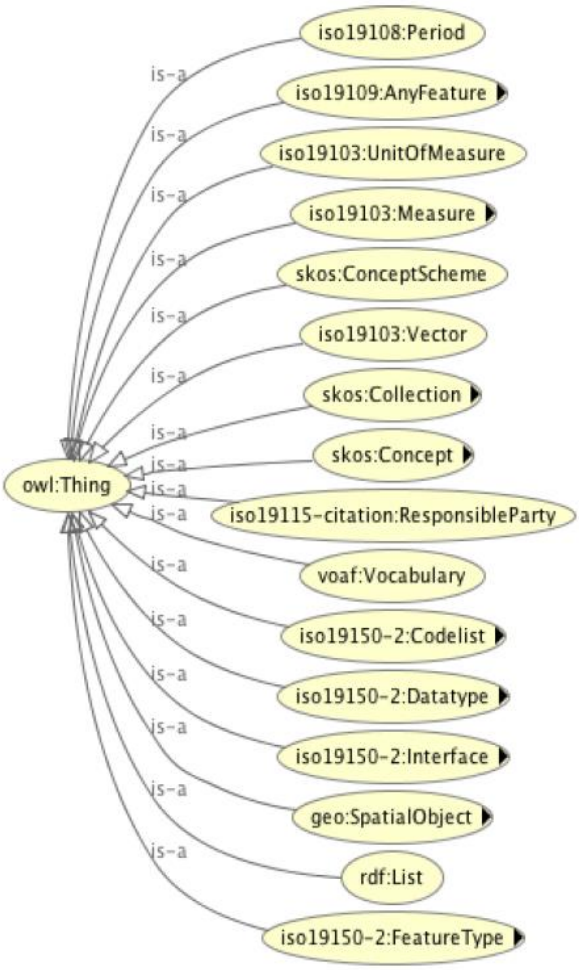


Figure 17. FOODIE ontology top hierarchy

The main motivation for the ontology was to represent a continuous area of agricultural land with one type of crop species, cultivated by one user applying one farming mode (conventional vs. transitional vs. organic farming). Such a concept is called Plot and represents the main element in the model, especially because it is the level to which the majority of agro data is related. One lower level than Plot is the ManagementZone, which enables a more precise description of the land characteristics in fine-grained areas. The Plot has two kinds of data associated:

(i) metadata information, including properties: code (id), validity (when the plot started and ceased to exist), geometry (spatial extent), description and originType (manual, system);

(ii) agro-related information, including:

- ProductionType, representing production-related data, having properties: productionDate (data of adding/changing of information in the knowledge base (KB)); variety (variety of crops);

productionAmount (quantity of variety)

- CropSpecies, representing the planted crop species, having properties: date (date of start/end of crops in a Plot); cropArea (spatial extent on a plot); cropSpecies (common species name of crop)
- Alert, representing alerts generated by the models integrated in the platform, having properties: code; type (according to user-defined classification); description; checkedByUser (indication of user awareness); alertDate (creation); alertGeometry (spatial extent of alert).
- Intervention, representing the basic feature type for any application with defined geometry, having properties: type; description; notes; status; creationDate (in KB); interventionStart/End (real time star/end); interventionGeometry (spatial extent of intervention); supervisor (authorized entity to supervise execution); operator (person who executed it); evidenceParty (entity who added it in the Knowledge Base); price.

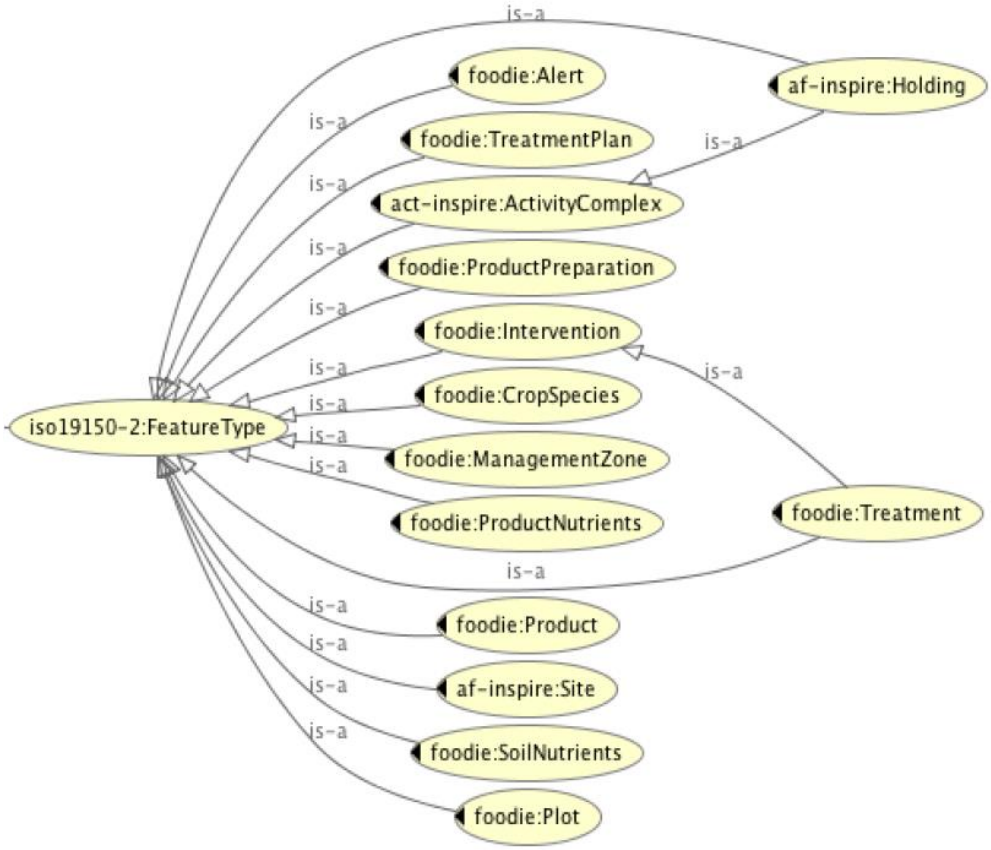


Figure 18. FOODIE ontology: ISO 19150-2 FeatureType class subclasses (=ISO 19109 AnyFeature & geosparql Feature classes)

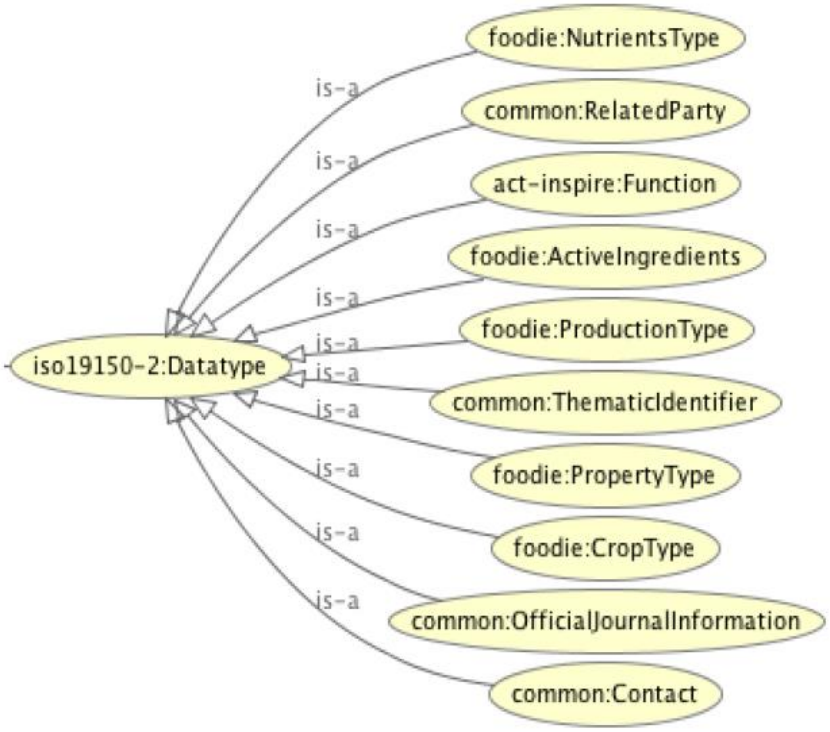


Figure 19. FOODIE ontology: ISO 19150-2 Datatype class subclasses

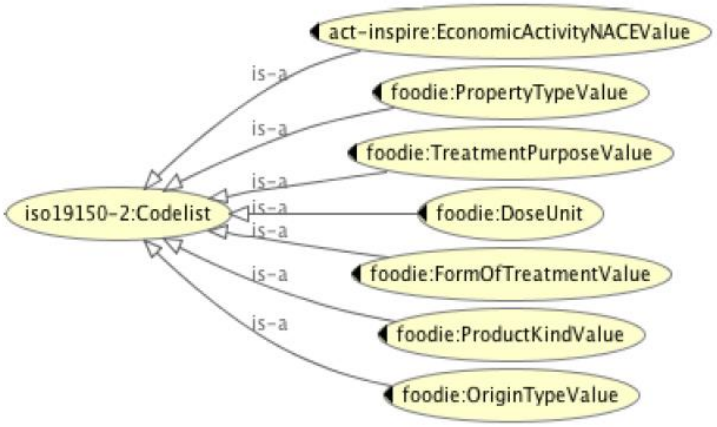


Figure 20. FOODIE ontology: ISO 19150-2 Codelist class subclasses

The intervention has direct and indirect associations to the following entities:

- Treatment comprising properties: quantity (applied physical quantity); tractorId (vehicle for machine applying it); machineId (machine applying it); motionSpeed (recommended speed for its application); pressure (recommended pressure for its application); flowAdjustment (indication if flow adjustment was

needed for its application); applicationWidth (width in which a machine is capable to apply it); areaDose (maximum application rate); formOfTreatment (id of its application, e.g., manual, aerial, from a code list); treatmentPurpose (rationale why it was used, e.g., weed, pest, from a code list); treatmentDescription.

- TreatmentPlan comprising properties: treatmentPlanCode; description; type; campaign (period to which it was designed); treatmentPlanCreation (in the KB); notes.
- ProductPreparation comprising properties: productQuantity (physical quantity of the applied product); solventQuantity (physical quantity of solvent applied); safetyPeriod (when a dissolved product may be used).
- Product, comprising properties: productCode; productName; productType (free text); productSubType (detailed classification); productKind (origin, e.g., organic, mineral - from a code list); description; manufacturer; safetyInstructions; storageHandling (for safe storage); registrationCode (id according to the national or other relevant registration scheme); registerUrl (link to the national (or other) registry); nutrients (id of nutrients, i.e., chemical elements and compounds necessary for plant growth, represented by NutrientsType class comprising properties for the amount of nitrogen, phosphorus pentoxide, potassium oxide and other chemical elements.
- ActiveIngredients with properties: code, ingredientName, and ingredientAmount.

FOODIE ontology has been used and refined in the DataBio project, and additionally, some extensions have been further defined to cover specific topics or application needs.

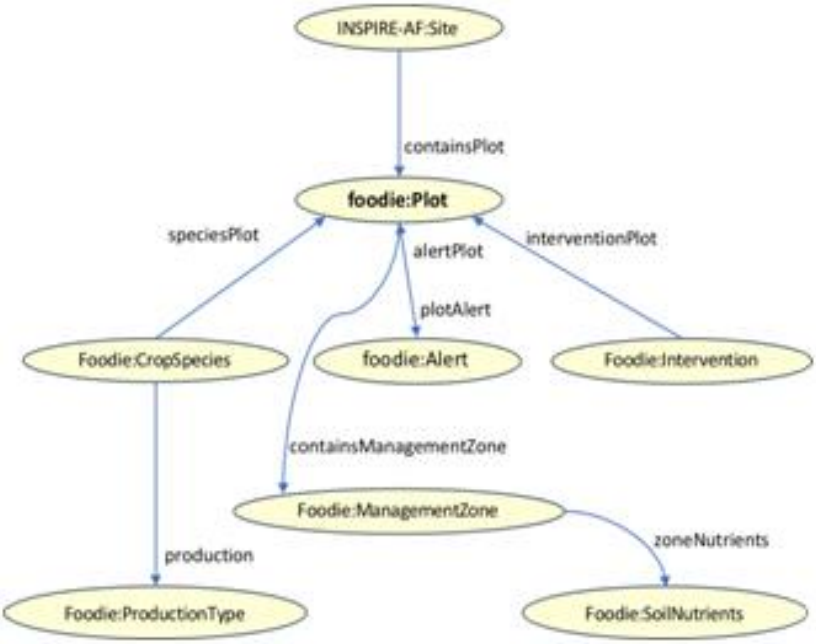


Figure 21. FOODIE ontology: Partial view of core classes and relations

5.11 FOODON Ontology

The FOODON[29] project aims to build a comprehensive and easily accessible global farm-to-fork ontology about food, that accurately and consistently describes foods commonly known in cultures from around the world. FOODON addresses food product terminology gaps and supports food traceability (following the GS1 standard). The FOODON ontology covers basic raw food source ingredients, process terms for packaging, cooking and preservation, and an upper-level variety of product type schemes under which food products can be categorized. It is built to interoperate with the OBO Library and to represent entities which bear a "food role".

As can be seen in Figure 22 below, we present a food product diagram based on the FoodOn ontology; this depicts several of the classes and relationships defined in it and contains information such as the source of food used to create the product, what kind of processing has been done in order to produce it, but also other information such as the packaging it is in, as well as cultural origins as well as consumer groups for it.

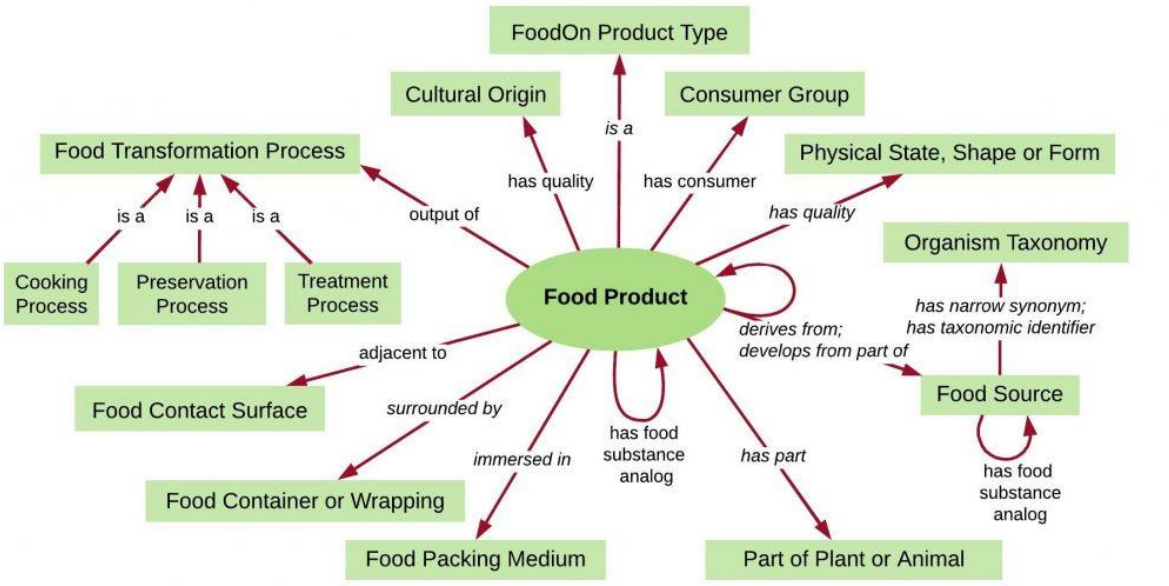


Figure 22. Food product diagram based on the FoodOn ontology depicting several of the classes and relationships defined in it.

More specifically, the ontology covers terms for the origin of food sources, the processing and cooking that a

[29] Dooley D.M. et al. "FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration" (2018), Nature; also see website: https://foodon.org/

product has undergone (e.g. chemical or heat processing to mention a few), even covering how it is packaged (the container or the wrapping of this product). Of course, it also includes information regarding where the material (e.g. meat or plant etc.) for the food originated, any additives needed for the processing as well as detailed information regarding the processing (e.g. in chemical treatment what types of chemical and additives are used, or to which degree it was heated during a heating procedure).

In Figure 23 below, we present an example usage of the FoodOn ontology when used to describe food product information for corn flakes showing the material from which they are made (i.e. corn which has then been processed to corn meal), that it's a type of cereal and that it has undergone milling and flaking processes and finally has been produced as the output of a dehydration procedure.

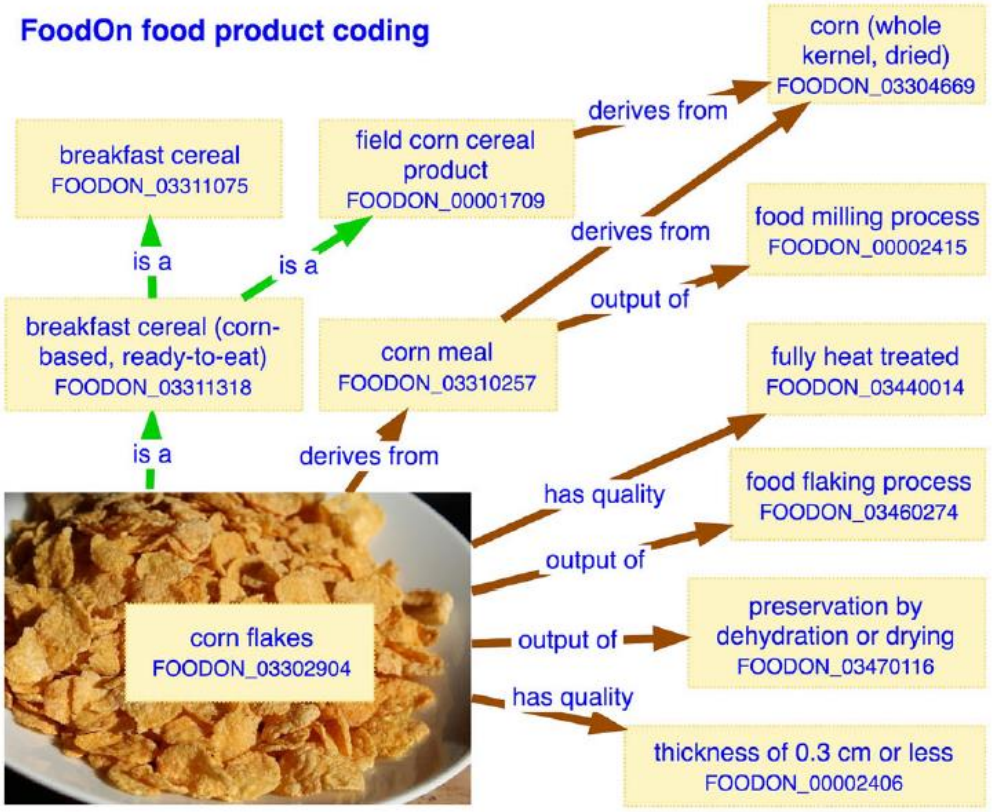


Figure 23. An example usage of the FoodOn ontology describing food product coding for corn flakes

This ontology generally categorizes the origins of each food source by defining named individuals for each country (and furthermore US state) in the world; however it might be more appropriate to use GPS or factory and farm information in order to further specify where each food source originates from or facilities where food processing took place.

5.12 Weather Data Models

These data models describe entities useful for dealing with weather data. These entities are primarily associated with the vertical segments of the environment and agriculture but is applicable to many different applications.

The main entities identified are:

- WeatherForecast: It represents a weather forecast for a period of time and a location.
- WeatherObserved: It represents a weather observation made over a period of time at a specific location.
- WeatherAlert: It represents a weather alarm intended to raise attention over a forecasted extreme weather condition.

5.12.1 WeatherObserved

It represents a weather observation made over a period of time at a specific location.

The Data Model has the following properties:

- id: Unique identifier.
- type: Entity type. It must be equal to “WeatherObserved”.
- dataProvider: Specifies the URL to information about the provider of this information.
- dateModified: Last update timestamp of this entity.
- dateCreated: Entity's creation timestamp.
- name: Name given to the weather observed location.
- location: Location of the weather observation represented by a GeoJSON geometry. Required if address is not present.
- address: Civic address of the weather observation. Sometimes it corresponds to a weather station address. Required if location is not present.
- dateObserved: The date and time of this observation in ISO8601 UTCformat. It can be represented by a specific time instant or by an ISO8601 interval. (required)
- source: A sequence of characters giving the source of the entity data.
- refDevice: A reference to the device(s) which captured this observation.
- refPointOfInterest: A reference to a point of interest (usually a weather station) associated to this observation.
- weatherType: The observed weather type (clearNight, sunnyDay, slightlyCloudy, partlyCloudy, mist, fog, highClouds, cloudy, veryCloudy, overcast, lightRainShower, drizzle, lightRain, heavyRainShower, heavyRain, sleetShower, sleet, hailShower, hail, shower, lightSnow, snow, heavySnowShower, heavySnow, thunderShower, thunder).
- dewPoint: The dew point encoded as a number.

- visibility: Visibility reported (veryPoor, poor, moderate, good, veryGood, excellent)
- temperature: Air's temperature observed.
- relativeHumidity: Air's relative humidity observed (percentage, expressed in parts per one).
- precipitation: Precipitation level observed.
- windDirection: The wind direction expressed in decimal degrees compared to geographic North (measured clockwise), encoded as a Number.
- windSpeed: The observed wind speed in m/s, encoded as a Number.
- atmosphericPressure: The atmospheric pressure observed measured in Hecto Pascals.
- pressureTendency: Is the pressure rising or falling? It can be expressed in quantitative terms or qualitative terms. (raising, falling, steady)
- solarRadiation: The solar radiation observed measured in Watts per square meter.
- illuminance: The illumninance observed measured in lux (lx) or lumens per square metre (cd·sr·m−2).
- streamGauge: The water level surface elevation observed by Hydrometric measurement sensors, namely a Stream Gauge, expressed in centimeters.
- snowHeight: The snow height observed by generic snow depth measurement sensors, expressed in centimeters.

5.12.2 WeatherForecast

It represents a weather forecast for a period of time and a location.

The Data Model has the following properties:

- id: Unique identifier.
- type: Entity type. It must be equal to “WeatherForecast”.
- dataProvider: Specifies the URL to information about the provider of this information
- dateModified: Last update timestamp of this entity.
- dateCreated: Entity's creation timestamp.
- name: Name given to the weather forecast location. (required)
- location: Location of the weather observation represented by a GeoJSON geometry. Required if address is not present.
- address: Civic address of the weather forecast. Required if location is not present.
- dateRetrieved: The date and time the forecast was retrieved in ISO8601 UTC format. (required)
- dateIssued: The date and time the forecast was issued by the meteorological bureau in ISO8601 UTC format. (required)
- validity: Includes the validity period for this forecast as a ISO8601 time interval. As a workaround for the lack of support of Orion Context Broker for datetime intervals, it can be used two separate

attributes: validFrom, validTo. required

- validFrom: Validity period start date and time.
- validTo: Validity period end date and time.
- source: A sequence of characters giving the source of the entity data.
- refPointOfInterest: A reference to a point of interest associated to this forecast.
- weatherType: The forecasted weather type.
- visibility: Visibility forecasted.
- temperature: Air's temperature forecasted.
- feelsLikeTemperature: Feels like temperature forecasted.
- relativeHumidity: Air's relative humidity forecasted (percentage, expressed in parts per one).
- precipitationProbability: The probability of precipitation, expressed as a number between $0 \leq$ precipitationProbability ≤ 1.
- windDirection: Wind direction forecasted.
- windSpeed: Wind speed forecasted.
- dayMinimum: Minimum values forecasted for the reported period.
- dayMaximum: Maximum values for the reported period.
- uVIndexMax: The maximum UV index for the period, based on the World Health Organization's UV Index measure.

5.13 Information Management Adapter (IMA)

In order to contribute to the overall efforts for building and extending the IoT ecosystem around Smart Farming (SF) in Europe, a FIWARE-powered solution is presented, called "Information Management Adapter" (IMA). It aims to enable interoperability at semantic and syntactic level for existing SF systems interoperation. The core design concept for the IMA is to be generic enough in order to be deployed on top of a targeted SF system, with minimum customisation efforts, to translate provided data streams to NGSIv2 data model and make them available to authorised parties through the NGSIv2 API in a secure and efficient manner. The IMA is also able to receive data from interoperable 3rd parties which further facilitates information richness for the underlying system. The IMA is expected to be deployed on the administrative cyber-premises of the underlying smart-farming solution; it will be deployed at or near the local network perimeter following a deployment approach similar to the Science Demilitarized Zone (ScienceDMZ). This approach ensures that the internal operations of the SF enterprise network will remain intact while any special security and data transfer policies can selectively be applied to the portion of the network designed to facilitate the transfer of interoperable data. The IMA is reusing the OCB for facilitating data exchange and Cygnus[30] for enabling data persistence on historic context

[30] https://fiware-cygnus.readthedocs.io/en/

information. Having established the necessary interoperability mechanisms that facilitates information richness and data homogenisation, the IMA also optionally provides context-aware decision support mechanisms based on agriculture scientific models (e.g. irrigation, fertilisation, pest management). This structure of IMA is depicted in Figure 24.

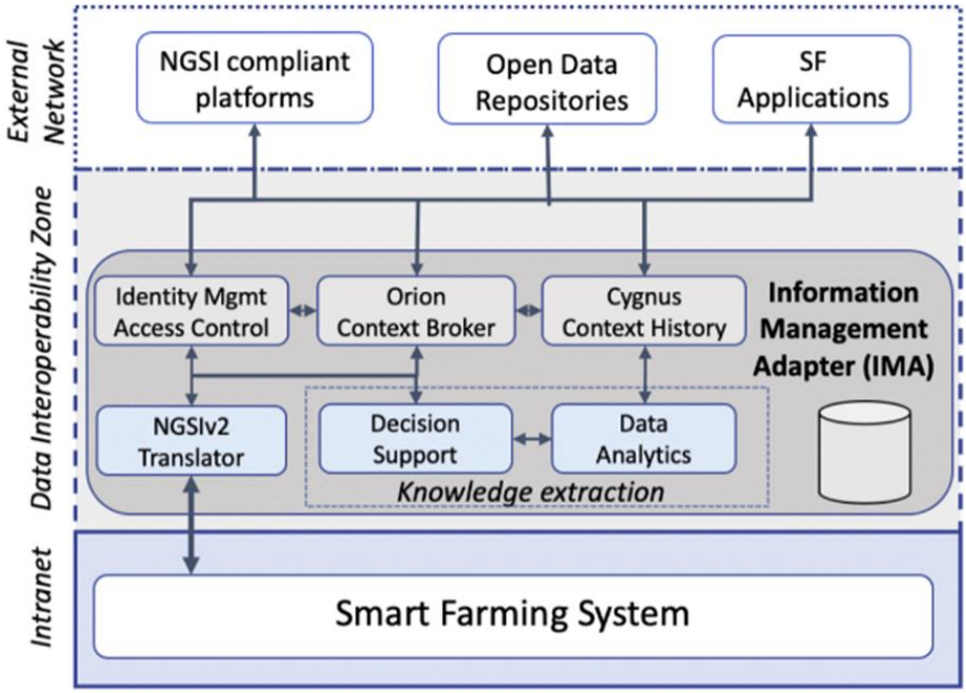


Figure 24. Information Management Adapter (IMA) connected to a Smart Farming System

5.14 ADAPT (Agricultural Data Application Programming Toolkit)

Different brands of farm equipment and software currently collect and consume data in a variety of proprietary file formats. Although this is a natural consequence of the industry's growth, it makes it hard for end-users to "assemble the dots" and output value from the data. This absence of interoperability in agricultural field operations is not just a problem of a deficiency of common data formats or syntax. There has also been a lack of a shared understanding, or semantics, among the different industry actors involved in field operations. This means using multiple terms or codes to refer to the same concept or using the same term to refer to multiple concepts.

A consequence of the previously described absence of common semantics is a lack of common Reference Data; i.e., a common set of code lists, unique identifiers and controlled vocabularies that can be used to identify resources such as crop inputs, farm machine, implement and sensor models, etc., in a consistent way understood

by all. In addition, growing public interest in sustainability, traceability, and agreement reporting need an ever-increasing amount of data to be collected as part of everyday operations in modern production agriculture. This requirement usually adds considerable amounts of frequently changing, geopolitical, context-dependent information such as identification numbers specific to the government agencies with whom the grower interacts with in their jurisdiction. Realizing all these requirements in the data model of Farm Management Information System (FMIS) software is very difficult, especially in an international context and given the realities of corporate information technology, where the release frequency of new software versions is constrained in many ways.

The keys to solving the previously described problems could be summarized as follows:

1. Decouple the infrequently and frequently changing aspects of FMIS data models.
2. Interoperate despite the multiple, often proprietary data formats used in the industry.
3. Develop a framework to capture and express meaning in field operations.
4. Develop a framework for sharing Reference Data across the industry.

AgGateway[31], a non-profit consortium of over 200 companies dedicated to the implementation of standards to advance digital agriculture, created its Precision Agriculture Council in 2011 in order to collaboratively tackle these interoperability problems. This led to the creation of the Agricultural Data Application Programming Toolkit (ADAPT) team, which is in charge of realizing a common object model for field operations as well as a set of format conversion tools. The ADAPT common object model meets requirements from AgGateway's SPADE (planting, crop care, harvest and post-harvest - scoped) and PAIL (irrigation, observations and measurements - scoped) projects, and also pursues compatibility with the ISO11783-10 standard XML format (ISO, 2015) and participant companies' own systems; for example, a Grower (which stands for the business entity, rather than the person, which is modelled using Person and PersonRole classes, not shown) has an attribute called Name, which is of the String class.

5.14.1 The ADAPT Object Model

An object model represents some part of the world that is of interest. Object models are composed of *classes* (which represent groups of related data; an object-oriented version of the classic concept of a *data type*) and the *relationships* among them. Figure 25 shows a small part of the ADAPT object model with five classes which are the boxes labelled Grower, Farm, etc. and the relationships among them (the arrows). The data (*attributes*) contained in each class are listed inside its box; for example, a Grower (which represents the business entity, rather than the person, which is modelled using Person and PersonRole classes, not shown) has an attribute called Name, which is of the String class.

[31] www.aggateway.org

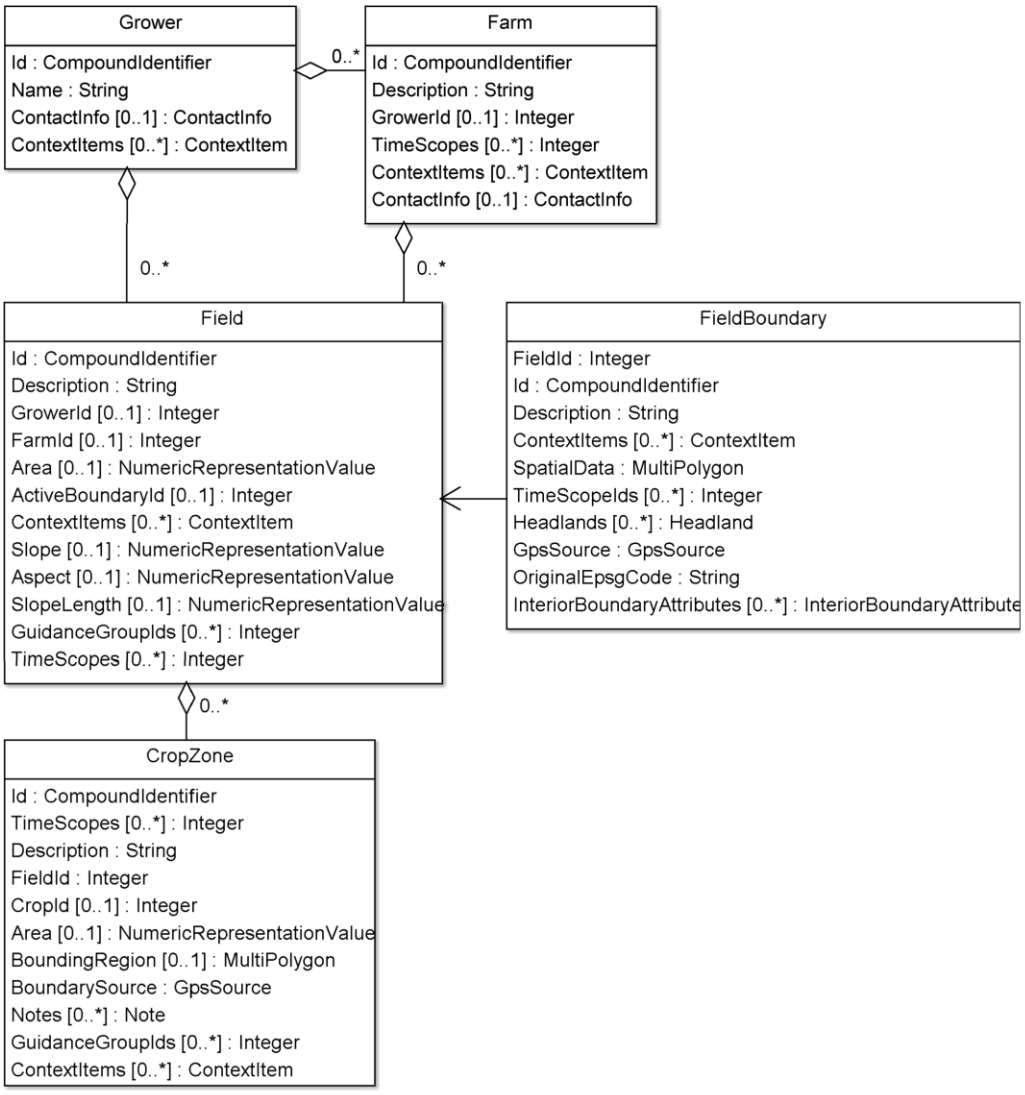


Figure 25. Fragment of the ADAPT object model

In this figure, simple arrows represent associations, for example, when an object contains references to another object. The arrows tipped with diamonds represent a more specialized form of an association called *aggregation*, where an object contains other objects; for example, a farm can have multiple fields, but these contained objects can "stand alone", and have their own identity independently of the containing object. The way in which the ADAPT team implemented some of these concepts, for example, fields being able to exist independently of farms, was a result of maintaining compatibility with the ISO 11783 standard, and with particular contributors' internal data models. In order to use the ADAPT data model, which is also called "application data model" or ADM, FMIS software must first create an instance of it, and then begin populating it with instances of objects as

needed by the application at hand. For example, if a grower wishes to use ADAPT to send a Work Order for a spraying application to a retailer or custom applicator, their FMIS would follow several steps:

- Create an instance of the ApplicationDataModel, and its children to the Catalog and Documents objects.
- Create objects for the *Reference Data* needed by the Work Order and put them inside the Catalog; for example, objects that describe the crop protection products that will be applied in the work order. Reference Data is meant to be universal, and independent of the specific grower.
- Create objects for the *Setup Data,* which are needed by the Work Order, and put them inside the Catalog. Setup Data refers to grower-specific data objects that describe resources being used by the document of interest, for example the Work Order, but that does not specifically describe the *state* of the resource. Examples include objects for the grower, farms, fields, crop zones, and boundaries that may be a target of the work order. If the work order was internal to the grower's own operation, it could also include references to specific machines, and to operators thereof.
- Create objects describing the desired field operation and put them inside the Documents object. In this case, it would need to create instances of:
    - The Work Order document.
    - A WorkItem object, representing the desired pass over the field and referenced by the Work Order.
    - A WorkItemOperation object that describes the spraying operation.
    - One of two possible Prescription objects (chosen based on the level of spatial detail desired to specify placement), that reference the Product objects created earlier and that describe how much of each product to implement to the fields and crop zones represented by previously created Setup Data.

An important aspect of ADAPT is how it manages identification. Figure 25 shows the use of a class called "Compound Identifier" to identify instances of objects. This class provides powerful functionality meant to reconcile different identification schemes used in the industry; its use was described in detail by AgGateway (2017).

5.15 Metadata Standards suitable for the agrifood sector

5.15.1 DCAT

The DCAT vocabulary expresses semantics related to meta-information about datasets. The W3C-recommended Data Catalog Vocabulary DCAT has been designed to describe datasets in data catalogs, i.e., in "curated collections of metadata about datasets". DCAT covers basic metadata about catalogs and datasets, as well as technical ways of distributing datasets (e.g., as a file for download or via a query API). DCAT-AP (Application Profile) has been specifically designed for public sector datasets; it specifies an extension of DCAT by further standard metadata vocabularies to obtain a more comprehensive description of datasets. Version 2 of the DCAT

recommendation is currently being drafted [DCAT20]; it has been evolved considering many of the DCAT-AP extensions.

5.15.2 W3C Data Quality Vocabulary, PROV-O and DUV

Capturing and processing and acting upon data quality information is an integral part of the DEMETER data management and integration process. The Data Quality Vocabulary (DQV) is an extension to the DCAT vocabulary which covers data quality, frequency of updates, whether the data accepts user corrections, persistence commitments, etc., and data provenance information [DQV16]. The figure below depicts the main classes of DQV.

Data provenance can be covered by the PROV-O Ontology [LeSa]. The PROV-O data model provides a set of classes properties and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts. DQV is re-used in the DQV to represent the provenance of the analysed data set and of the quality metadata.

An additional standard, re-used by DQV, is the Data Usage Vocabulary (DUV) [LoSt]. This standard is used to describe consumer experiences, citations, and feedback about the dataset from a human perspective. Information relevant for DEMETER, such as, fitness for purpose, adoption and reliability, and usability and openness can be expressed via *duv:Usage* and *duv:UserFeedback* classes. Those concepts, however, are rather generic and might require further extension to be able to express the required concepts more explicitly.

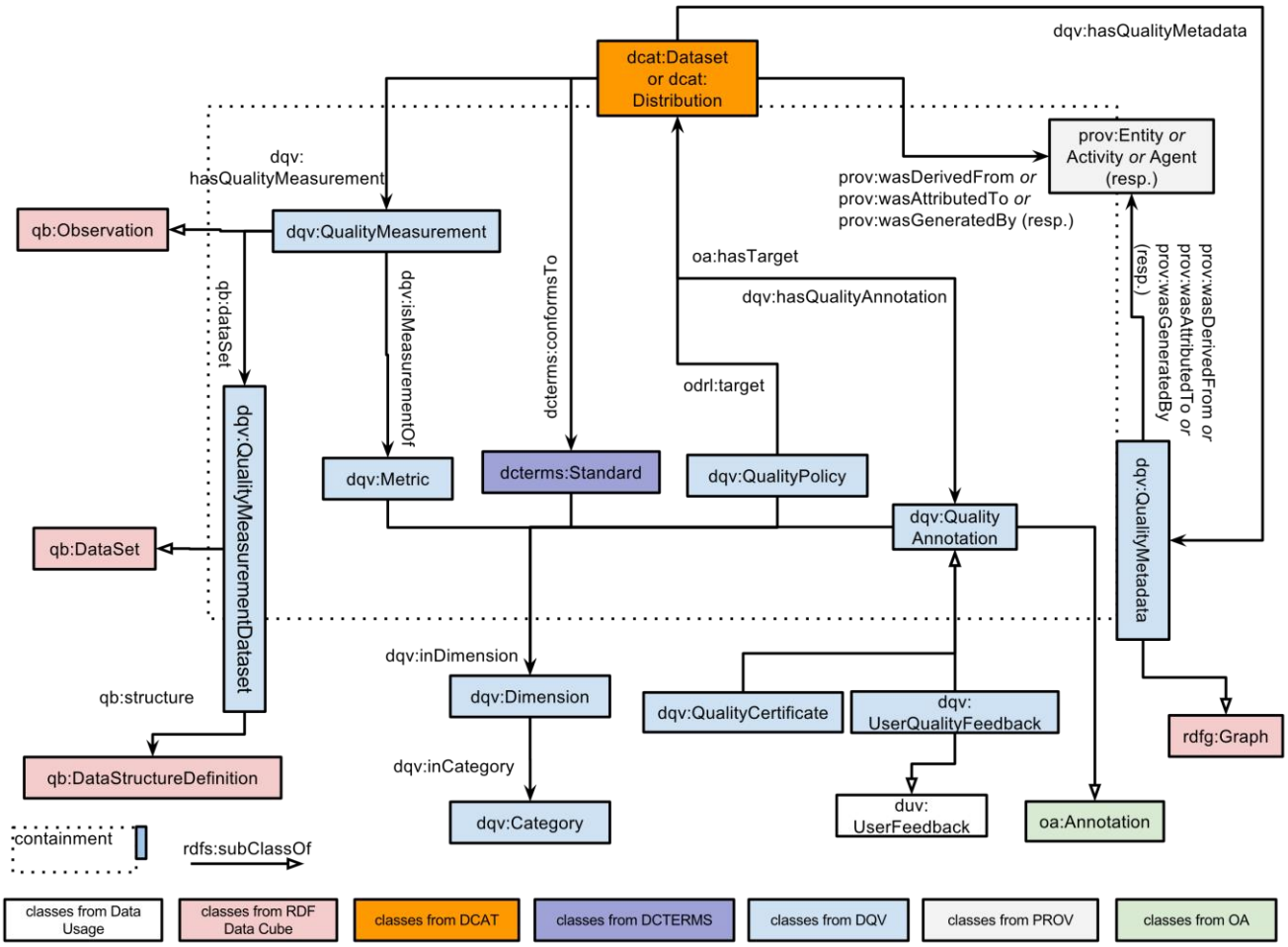


Figure 26. Data Model of the W3C Data Quality Vocabulary

5.15.3 IDS Information Model

The IDS Information Model [IDSA20], [OtSt19] is an RDFS/OWL-ontology which covers fundamental concepts of the International Data Spaces, i.e. types of digital contents that are exchanged by IDS participants via IDS infrastructure components. The challenges addressed by IDS, including interoperability and security, are related to those of DEMETER, and it could make sense to consider an adoption of IDS concepts or particular components. The IDS Information Model has been realized as an extension of DCAT 1 [DCAT20] which incorporates standards like DQV [DQV16], ODRL [ODRL18] and related standards for the description of data resources. The model has been developed in an exchange of ideas with the working group that specified DCAT.

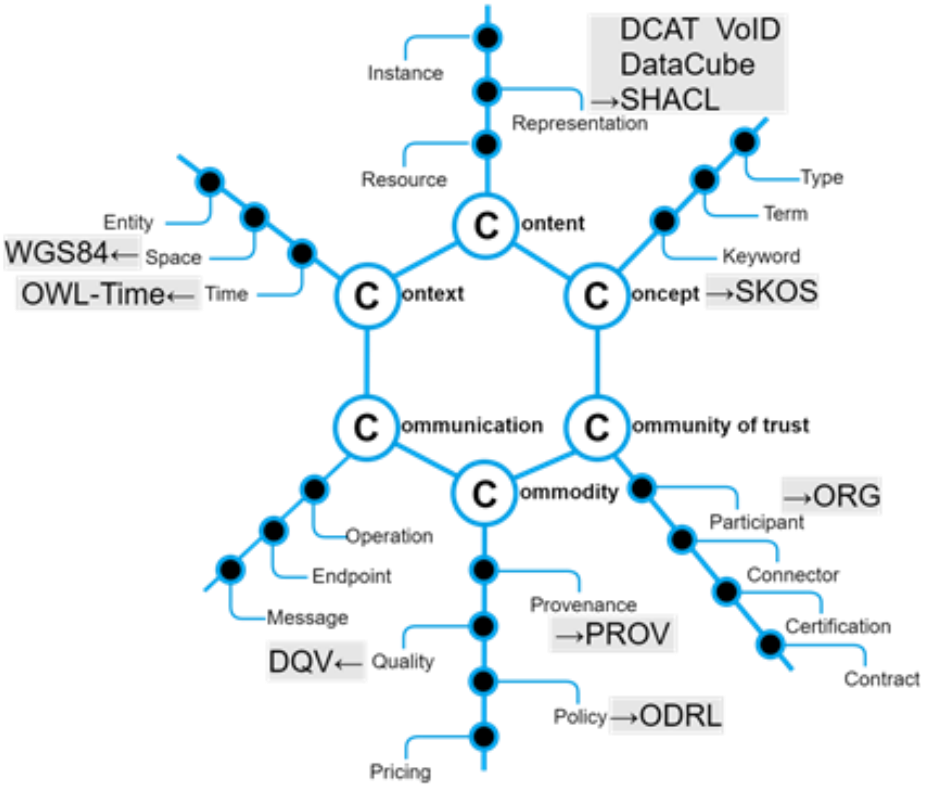


Figure 27. The "Concern Hexagon" of the IDS Information Model enhanced with references to standards reused

From an NGSI-LD perspective, metadata is meant to convey high-level information about contents of datasets (data items), while the IDS Information Model also captures contextual properties of data sets or, more generic, digital resources. As clarified in the Figure below, it covers six broad concerns of sharing digital resources, as well as the detailed aspects of these concerns. In addition to the mere meta level, the IDS Information Model provides extension points to also convey information about the (usually domain-specific) structure and semantics of the content of datasets. To this end, it reuses the W3C standards VoID (for pointing to domain ontologies and their terms used in a dataset), Data Cube (for describing the structure of tabular or matrix-like datasets) and, in the most general case, SHACL [ShaCL17], for describing general graph-shaped structures of datasets (cf. examples at [ShaCL20]).

The VoID Vocabulary of Interlinked Datasets[32] has been proposed by W3C for expressing metadata about RDF datasets. In comparison to DCAT, which does not specifically address RDF, it enables an RDF-specific representation of information on:

1. how to access a dataset (e.g., what resources, i.e., nodes are in an RDF graph, are suitable starting

[32] https://www.w3.org/TR/void/

points for exploring a dataset),
2. how a dataset is structured (e.g., what URI patterns it uses, what vocabularies and what terms from them it uses, what subsets in terms of, e.g., instances of certain classes or properties it has, etc.),
3. how a dataset is linked to other datasets.

Regarding points (2.) and (3.), VoID is more expressive than the core of the IDS Information Model. Therefore, the specification of the IDS Information Model recommends reusing VoID for these topics.

5.16 Semantic Interoperability mechanisms for the agrifood sector

Having presented in the previous sections, two specific solutions (IMA and ADAPT) that promote semantic interoperability between the ontologies and data models that are used in the agrifood sector, we present now in this section a more general overview of semantic interoperability mechanisms. Then, we present an overview of how to achieve interoperability across data from different sources via translation into a common, interoperable format. Finally, we present two specific mechanisms that follow this paradigm.

5.16.1 Interoperability definition and levels

According to the IEEE Glossary: “Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged”.

Interoperability needs to be considered at three levels: [EIF04]

- The *organizational level*: coordinated processes in which different organizations achieve a previously agreed and mutually beneficial goal (this goal could pertain to satisfying desired policies).
- The *semantic level*: the precise meaning of exchanged information, which is preserved and understood by all parties, which includes the desired behaviour of assets (agents, machines, systems) and data exchanged.
- The *technical level*: planning of technical Issues involved in linking computer systems and services, part of which is to achieve syntactic interoperability and to define the communication transport between assets.

This organisation is depicted in Figure 28 taken from the IEC white paper [IEC19]:

Figure 28. Generic semantic interoperability scenario

Organisational interoperability is concerned with how organisations cooperate to achieve their mutually agreed goals. In practice, organisational interoperability implies integrating business processes and related data exchange. Organisational interoperability also aims to meet the requirements of the user community by making services available, easily identifiable, accessible, and user focused.

Semantic interoperability allows organisations to process information from external sources in a meaningful manner. It ensures that the precise meaning of exchanged information is understood and preserved throughout exchanges between parties. To achieve semantic interoperability, it is necessary to define the data structures and data elements for the given application domain and agree on the meaning of the information to be exchanged. This should be the goal of any information management system.

Regarding *technical interoperability,* we should consider foremost *syntactic interoperability*; this is a key aspect of technical interoperability and a prerequisite for semantic interoperability. The difference between semantic and syntactic interoperability is as follows. Semantic interoperability is about the meaning of data elements and the relationship between them. It includes developing vocabularies to describe data exchanges. Syntactic interoperability is about describing the exact format of the information to be exchanged in terms of grammar, format, and schemas. In practice, the transition is not clear cut. Where schemas enable syntactic interoperability, part of the semantics may be represented in an explicit, machine-comprehensible and machine-executable way,

thus turning the schema into a vocabulary.[33] When even more semantic knowledge is being captured using formal logic (axioms and/or rules), one also speaks of ontologies.

Further aspects of *technical interoperability* cover the technical aspects of linking information systems. It includes aspects such as interface specifications, interconnection services, data integration services, data presentation and exchange, etc.

The graph-based RDF data model, together with the RDF Schema (RDFS) and the Web Ontology Language (OWL), on which most of the standards mentioned above in the context of semantic (conceptual) data models are based, is a technical foundation that is particularly well suited for enabling semantic interoperability, because further information, even if incomplete, can be attached to any existing node in a graph, and because the same structure is used for coherently representing all of the three following aspects of data:

- the *data itself*: the ground truth as, e.g., measured by sensors.
- the *metadata about the data*, which include:
    - the context (space, time, units of measurement) in which the data was obtained[34],
    - further information about provenance including the agents who created data or the processes from which data resulted[35],
    - further information about social interaction with data[36],
    - licenses and policies that restrict the reuse of data[37].
- the *schema of the data*, including:
    - syntactic and semantic integrity constraints,
    - the structure of data, e.g., as a multi-dimensional matrix[38],
    - capturing background/expert knowledge about the application domain.

5.16.2 Achieving interoperability by translation into interoperable formats

[33] For example consider http://schema.org/, a community standard that captures the semantics of things commonly searched for on the Web on a basic level.
[34] Context is they key notion of the NGSI-LD information model; NGSI-LD is presented in an earlier section of this report (see Section 5.2).
[35] Addressed, e.g., by the W3C PROV standard (https://www.w3.org/2001/sw/wiki/PROV)
[36] Addressed, e.g., by the RDF Review Ontology or various tagging ontologies including MUTO (see http://muto.semantic-interoperability.org/core/v1.html)
[37] Data usage policies can be defined by the W3C Open Digital Rights Language (ODRL) (see https://www.w3.org/ns/odrl/2/ODRL20.html), which has been extended by the International Data Spaces (IDS) Information Model. For IDS visit https://www.internationaldataspaces.org/ and for the latest visualisation of the IDS ontology see http://ids.semantic-interoperability.org/webvowl/index.html#ontology.
[38] addressed by the W3C Data Cube Vocabulary (see https://www.w3.org/TR/vocab-data-cube/).

Generally, interoperability across data from different sources is achieved by a physical or virtual translation into a common, interoperable format. This section provides a high-level summary of what the SotA on Data Management and Integration discusses in detail in the section on data integration approaches. This is commonly achieved by putting wrappers around the respective data sources, such that the wrappers access the data sources via their native access interfaces (e.g., via query languages such as SQL, or via web service APIs), and translate the source data to a common schema/vocabulary/ontology, ideally also eliminating duplicate representations of the same real-world object in different data sources or making explicit links between related real-world objects found in different data sources. There are two different approaches to translation, depending on the requirements:

Table 1. Approaches for translation into interoperable formats

Translation approach	Extract-Transform-Load (ETL)	Query rewriting
How it works	Extracting the data from the source(s), transforming it into a physical dataset in the target model, and loading the latter into a database system that supports the latter model (e.g., a triple store for RDF)	Accepting queries in the language of the target model (e.g., SPARQL for RDF), detecting (in case of multiple sources) the data source required for answering the query or part of it, rewriting the query into the native query language / API of the data source, rewriting the source's result set into the target model. In other words, a *virtual* dataset is created.
Overall pros and cons	High throughput, high latency	Low latency, low throughput
Most useful when	Source data is updated rarely but queried frequently	Source data is updated frequently but queried rarely

5.16.3 Examples of Semantic Interoperability mechanisms for the agrifood sector

For the agrifood sector specifically, there are several initiatives to address semantic interoperability: we present two key ones here: AgroXML and ISOBUS.

5.16.3.1 AgroXML

agroXML is a standard for representing and describing farm work. It is the result of a tight cooperation with producers of agricultural software and online service providers. It can be used within farm management information systems as a file format for documentation purposes but also within web services and interfaces

between the farm and external stakeholders as a means to exchange data in a structured, standardized and easy to use way. [Agr]

It provides elements and XML data types for representing data on work processes on the farm, including accompanying operating supplies like fertilizers, pesticides, crops, etc.

Now, agroRDF[39] addresses the semantic interoperability requirements by providing a semantic model based on agroXML. The class diagram of the AgroRDF ontology is presented in Figure 29.

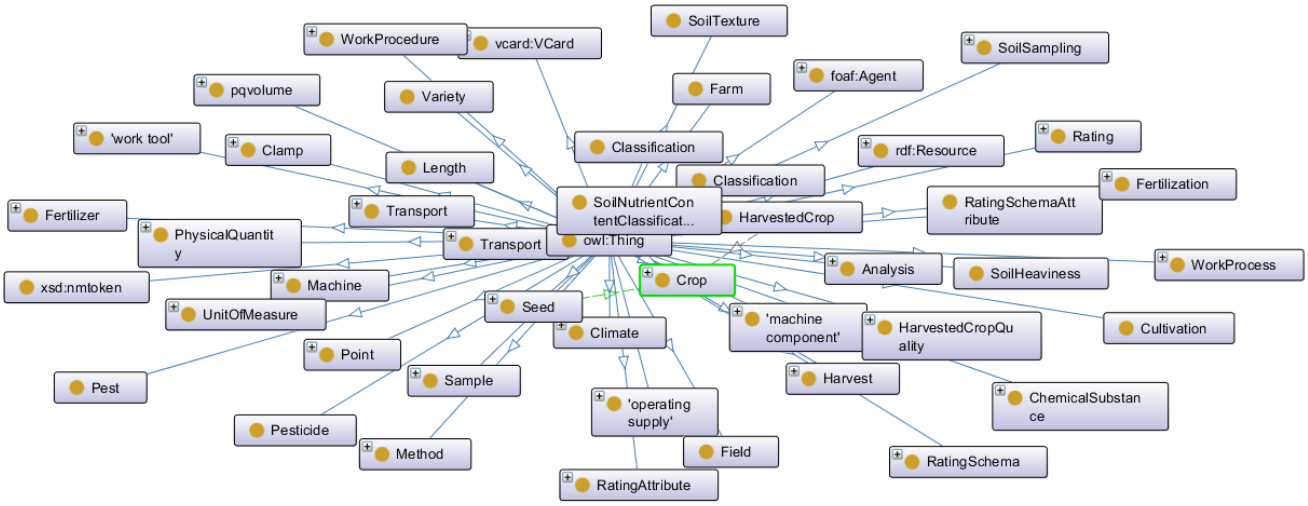


Figure 29. Class diagram of the AgroRDF ontology

5.16.3.2 ISOBUS and ISO11783-XML

The aim of ISOBUS[40] is to grant the syntactic interoperability between tractors and implements from different manufacturers, defining a standard protocol and semantics for agro functionalities. It offers a plug and play solution: only one terminal for a large selection of implements, regardless of the manufacturer.

All signals, such as speed, position of the lower links, power take-off RPM, etc. are available in standardized form for each implement. The communication between the implement and the farm management system is also standardized and simplified by using ISO-XML.

The validity of an ISO-XML file is defined by the schema ISO11783_TaskFile, that can be found at https://www.isobus.net/isobus/attachments/files/ISO11783_TaskFile_V2-1.xsd

[39] AgroRDF: http://data.igreen-services.com/agrordf
[40] https://www.aef-online.org/fileadmin/user_upload/Content/pdfs/AEF_handfan_EN.pdf

5.16.4 Semantic Definition Standardization

The semantic component of interoperability involves a lot of different contexts and cannot be realised in a single monolithic data model – such a model will not be interoperable across domains. Such models need to be built from standardised building blocks and applied to the application domain. This can be illustrated in the standardization strategy recommended by the IEC (see Figure 30).

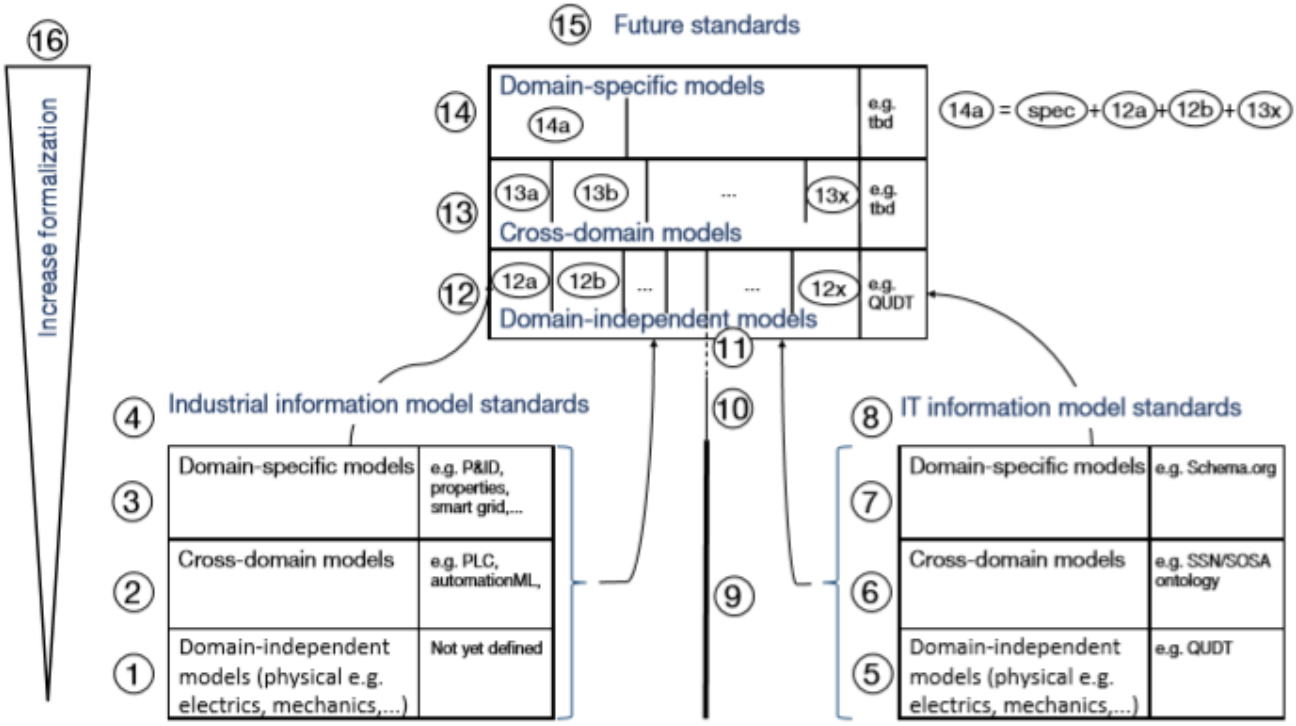


Figure 30. IEC strategy for building interoperable data models

Such a strategy is consistent with the identified best practices described in this document. The challenges are the level of detail of each model required and the "glue" between components.

Summary / Categorization of models

There are multiple different types of semantic components required to describe data models, and each has a

specific function. A summary of the main of these types is presented in Table 2.

Table 2. Model types, functions and examples of data models

Model type	Function	Examples
Meta-model	• allows different model components to be described in compatible ways • supports generic encoding	• RDF • NGSI-LD metamodel • OWL • RDFS
Class model	• defines the types of entities that are identified • defines relationships	• SKOS model • SOSA model • (OWL ontologies generally)
Schema	• Defines data exchange structures • Define expected properties of class instances	• XSD schema • FIWARE Agrigfood models
Dimension	• Defines semantics of data elements as independent or measured variables	• SDMX • RDF-Datacube
Controlled Vocabulary	• Defines the meaning of terms	• AGROVoc resources
Profiles	• Defines schema constraints • Defines vocabulary usage (allowable terms)	

A great deal of variability exists in the level of abstraction of models and schemas. These models may be interoperable, depending on the meta-model used and the class models expressed using these.

6 Technical Requirements

This section presents the technical requirements extracted by Task 2.1, which drive the data that the DEMETER enabled apps will need to use and therefore these will drive the development of the ontology used in the DEMETER AIM and in particular the domain specific ontologies developed (these will be presented in section 7.3). This is an exhaustive list of specific technical requirements that the DEMETER data model needs to be able to represent as well as requirements regarding the interoperability with existing systems and ontologies as well as requirements regarding the mapping of these data models to the DEMETER AIM. These two separate classes of requirements are presented in the next two subsections. This template will be extended by an additional field to capture the relevant Use Cases, once the pilot use cases are finalised.

6.1 Data Model and Data Modelling

Requirement ID	DK1.1	**Version**	0.2	**Last Update Date**	13/12/2019
Title	Common data view for heterogeneous models				
Description	DEMETER needs to define a common data model that defines a common view on all heterogeneous entities connected and all the data involved in the pilots. This common data model shall be used for all data exchanged between software components. Therefore, it needs to support the translation of the obtained data streams to a common data model.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance existing (and if necessary introduce new) Information Models Objective 2: Build knowledge exchange mechanisms				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Domain experts, pilot leaders, semantic technologies experts				
Prerequisite(s)	Pilots' requirements				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	ICCS, TECNALIA, PSNC				

Requirement ID	DK1.3	**Version**	0.2	**Last Update Date**	04/12/2019
Title	Representation of crop farms data				
Description	DEMETER's data model needs to enable the common representation of agronomic data (e.g., crops, sensor data from the field, thermal/multispectral imagery from				

	UAVs, production data, geolocation data, planting data, irrigation logs, fertilisation logs, spraying logs, …) including: • Farm and economics modelling: agricultural type and economic size, production volumes and types, calculations according to results, etc. • Field data modelling: location and geometry of the field, planting date, planting distance, detailed yield information. • Field status modelling: e.g., water- or nitrogen-stressed fields, appropriate evaluation indices (e.g., Normalized Difference Moisture Index (NDMI)), need for fertilising. • Soil data modelling: soil temperature and moisture, soil physical and chemical analysis • Crops, treatment and fertilisation modelling: crop type, crop developing stages, crop cultivar or variety, crop health status and pests, pesticides, nitrogen levels, information from counting devices used for the control of insects or plagues. • Traceability information of crops (production, transport, retail) to be used in the product passport information • Water management modelling: water and energy consumption, water quality (e.g., salinity levels)
Relevant Pilot(s)	• Farm and economics modelling: 2.4 • Field data modelling: 3.4 • Field status modelling: 1.3, 3.1, 5.2 • Soil data modelling: 1.4, 3.2 • Crops, treatment and fertilisation modelling: 1.3, 1.4, 2.2, 3.1, 3.2, 3.3, 5.1, 5.3 • Traceability information of crops: 5.1 • Water management modelling: 1.1, 1.2, 1.3, 1.4, 3.1, 3.2
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	8: Unified agriculture ontology 17: Water Management Model
Involved stakeholders/actors	Solution providers, standardisation organisations

Prerequisite(s)	Data models should be based on existing ontologies where possible
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS, TECNALIA, UMU

Requirement ID	DK1.4	**Version**	0.2	**Last Update Date**	30/1/2020
Title	Earth Observation Data Representation				
Description	DEMETER needs to enable the representation of current earth observation (EO) data as well as historical EO data, including for example satellite data, remote sensing imagery, soil maps, vegetation indices, such as NDVI, EVI, NDRE, NDMI. It needs to also get EO metadata, e.g., through interfaces compliant with the OGC 13-026r8[41] specification.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models				
Relevant Innovation(s)	8: Unified agriculture ontology 4: Earth Observation data service				
Involved stakeholders/actors	Data consumers (DSS) and processors, Data publishers, system architects, data discovery agents				
Prerequisite(s)	Use Cases for EO data. EO data availability.				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	ICCS				

[41] http://www.opengis.net/doc/is/opensearch-eo/1.0

Comments/Remarks	"Earth Observation Data" in this sense means remote observation of the earth, as opposed to in-situ sensors. EO data measures distribution of a phenomenon in a spatial field. There are a wide range of sensors, and a wide range of intermediate processed products, where processing may be performed on georectification, image (mosaic) aggregation, temporal averaging, cloud cover filtering, image enhancement, spectral filtering, feature detection, etc. The variability of the descriptions required and systems performing these tasks leads to a high degree of potential for interoperability challenges without a disciplined common approach. OGC provides a suite of inter-related standards for EO data encoding and provision via services; however, the semantic description aspects will require additional design work. This needs to be done in the context of a standards oriented meta-model that informs the DEMETER implementations so that consistency of approach can be achieved both within DEMETER and across other domains. OGC EO models, W3C Semantic Sensor Network and RDF-Datacube and other building blocks for this meta-model need to be adopted, adapted or mapped to, in order to maximise the long-term value of DEMETER and allow re-use of software components.

Requirement ID	DK1.5	**Version**	0.2	**Last Update Date**	27/01/2020
Title	Representation of livestock data				
Description	DEMETER's data model needs to enable common representation of livestock data and traceability of products including: • Modelling of dairy & beef farms and data from farm robots: milk and meat production and quality, milk properties and quality (fats, proteins, somatic cells and bacterial content), economic data • Modelling of data from cows' wearables: animal ID, location, temperature, pedometer data, movement. • Modelling of animals' welfare, behaviour and habits: eating habits, respiration monitoring data, rumination, activity, rectal temperature control data, feed and water consumption data, biomarkers related with animal well-being and welfare (e.g., cytokine markers) • Food traceability information of dairy products and pastries (tracking of ingredients and supply chain). • Modelling of poultry farms: animal welfare, habits, living conditions, stress				

	levels, medical treatment, feeding patterns, feed origin. • Traceability information of poultry products (production, transport, retail) to be used in the product passport information • Modelling of apiary and hives: location of hives, apiary weight
Relevant Pilot(s)	• Modelling of dairy & beef farms and data from farm robots: 4.1, 4.2, 4.3, 5.2 • Modelling of data from cows' wearables: 4.1, 4.2, 4.3, 5.2 • Modelling of animals' welfare, behaviour and habits: 4.1, 4.2, 4.3, 5.2 • Food traceability information of dairy products and pastries: 5.2 • Modelling of poultry farms: 4.4, 5.4 • Traceability information of poultry products: 5.4 • Modelling of apiary and hives: 5.3
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	Innovation 8: Unified agriculture ontology Innovation 11: Data integration across the entire dairy supply chain
Involved stakeholders/actors	Solution providers, standardisation organisations
Prerequisite(s)	Data models should be based on existing ontologies
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS, TECNALIA, ENG

Requirement ID	DK1.7	**Version**	0.1	**Last Update Date**	30/1/2020
Title	Representation of meteorological and open spatial data				
Description	DEMETER needs to enable representation of weather data (e.g., temperature, humidity, wind speed/direction, solar radiation, pressure, etc.) and open spatial data				

	modelling. Meteorological data will be collected by interfacing with existing sensors, or new sensors that will be provided for this purpose.
Relevant Pilot(s)	1.4, 2.2, 3.1
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	8: Unified agriculture ontology 4: Earth Observation data service
Involved stakeholders/actors	Solution providers, standardisation organisations
Prerequisite(s)	Data models should be based on existing ontologies and standards
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS, TECNALIA

Requirement ID	DK1.8	**Version**	0.2	**Last Update Date**	11/12/2019
Title	Representation of agricultural machinery data				
Description	DEMETER needs to enable common representation of agricultural machinery data such as: • engine data • fuel consumption, • emissions • exhaust gas • NOx-conversion • exhaust temperatures The data are defined by Controller Area Networks Bus (CAN) protocol specifications, consequently it will be necessary to take into account that the translation of CAN-Bus Model into DEMETER Data Model, involves understanding the specific CAN Bus				

	information (the message set for subsystem data exchange) of the supplier to the vehicle subsystem into new information according to the new DEMETER Data Model communication specifications. The new Data Model needs to represent entities types and formats, relationships among them, possible range between the values (if any).
Relevant Pilot(s)	2.1, 2.2
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	3: Agricultural automation and control 8: Unified agriculture ontology
Involved stakeholders/actors	Solution providers, standardisation organisations
Prerequisite(s)	Data models should be based on existing ontologies
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	TECNALIA

Requirement ID	DK1.9	**Version**	0.2b	**Last Update Date**	30/1/2020
Title	Representation of farmer's preferences and DSS recommendations to them				
Description	DEMETER needs to enable common data model able to interpret farmers' needs and preferences including: • farmers' needs related to cost optimization (e.g. linking economical aspects of wholesale and retail prices), production issues (better quality of their products, crop variety per field, optimal date for planting and harvesting), cost/benefit analysis of field operations (irrigation/fertilization), optimization on irrigation/fertilization strategies, disease monitoring, yield analysis (e.g. the estimation of crop yield according to climate conditions), animal welfare tracking;				

	• production preferences (e.g. the use of non-chemical pesticides, attention to animal welfare, transparency to the consumers); • any other relevant data input collected during farm operations (related to animal welfare, crop production, product's characteristics). DEMETER should also enable common representation of recommendations and notifications to farmers, as well as the metadata used for providing recommendations to farmers through the DSS system and analytics tools. In this way, farmers' needs and preferences will be adequately analyzed (data integration and analysis) and decision support (visualization) will be provided. Moreover, the data model to be defined will have to provide the optimization of existing DSSs, allowing them to be used by other pilots, to increase the interoperability between the Pilots through the use of a common language and syntax, to identify the entities involved, the needed relationships and attributes to define the pattern schema of the model.
Relevant Pilot(s)	**1.1, 1.2, 2.2, 2.3, 2.4, 3.2, 3.3, 3.4, 4.2, 5.3**
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	8: Unified agriculture ontology 3: Agricultural automation and control
Involved stakeholders/actors	Farmers, dairy producer, solution providers
Prerequisite(s)	None
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS

Requirement ID	DK1.11	**Version**	0.2	**Last Update Date**	30/1/2020

Title	Flexible and extensible model representation
Description	DEMETER needs to support flexibility and extensibility in the representation of AIM through the use of a modular approach, the reuse or alignment with thesauri/classifications available as linked data, the use of property graphs and semantics, the use of appropriate data interchange models (e.g., RDF), knowledge representation languages (e.g., SKOS, RDFS, OWL) and rule languages (e.g., SWRL or OWL-RL), which would enable the semantic querying of data.
Relevant Pilot(s)	**ALL**
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	8: Unified agriculture ontology
Involved stakeholders/actors	Ontology engineers, semantic technologies experts
Prerequisite(s)	Competency questions, pilots' requirements
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS, PSNC, ENG, FhG.FIT, TECNALIA, OGCE,

Requirement ID	DK1.13	**Version**	0.2	**Last Update Date**	06/12/2019
Title	Representation of data quality metrics				
Description	DEMETER needs to include quality metrics in its data model. This data will be used for evaluating the accuracy, precision, granularity, completeness, consistency, timeliness, validity, uniqueness (where applicable) of the agrifood data and will be used by the data fusion and analytics tools that Demeter provides.				
Relevant Pilot(s)	**ALL**				

Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models
Relevant Innovation(s)	8. Unified agriculture ontology
Involved stakeholders/actors	Solution providers
Prerequisite(s)	DK1.1, Data/information for implementing metrics
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS

Requirement ID	DK1.14	**Version**	0.2	**Last Update Date**	11/12/2019
Title	Provide a basis for data exchange across stakeholders				
Description	DEMETER needs to enable data exchange across authorised stakeholders. To facilitate this, it needs to include data regarding the supply and usage of agri-data and any other type of data that is stored in the DEMETER unified ontology including any economic transactions regarding the usage of such data.				
Relevant Pilot(s)	All				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models Objective 2: Build data knowledge exchange mechanisms				
Relevant Innovation(s)	8. Unified agriculture ontology 9. Secure Agricultural data sharing services				
Involved stakeholders/actors	Authorised stakeholders				

Prerequisite(s)	Gathered data
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS

Requirement ID	DK1.15	**Version**	0.2b	**Last Update Date**	27/01/2020
Title	Data Models enabling analysis of large heterogeneous data				
Description	DEMETER needs to provide data models that will enable the analysis and processing of large amount of heterogeneous data, including their storage and transfer. DEMETER should take advantage of numerous sources, like wireless sensor networks and imagery to store, fuse and process all the data that are required by Demeter applications; see DK1.3-9 for details.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Pilot leaders, domain experts, ontology engineers				
Prerequisite(s)	Competency questions, pilots' data requirements				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	ICCS, PSNC, ENG, FhG.FIT, TECNALIA, OGCE				

Requirement ID	DK1.16	Version	0.2	Last Update Date	30/01/2020
Title	Model for Data Brokerage Services				
Description	DEMETER needs to provide a common information model for Data Brokerage Services. It is necessary to have enough data in order to describe the offered resources and Demeter-enabled entities, their capabilities as well as the policies of those that offer them including pricing policies. In addition, once such entities are contracted it is necessary to keep transaction and (potentially) whole supply chain information in the model. These could even include the representation of (fragmented) supply chain stakeholder information.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models Objective 2: Build data knowledge exchange mechanisms				
Relevant Innovation(s)	1. Agriculture Interoperability Space 8. Unified agriculture ontology				
Involved stakeholders/actors	Consumers, producers				
Prerequisite(s)	-				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	LESP				

Requirement ID	DK1.17	Version	0.2	Last Update Date	30/01/2020
Title	General Model for Interoperability				
Description	DEMETER needs to provide a general model for data interoperability, which should be flexible and extensible for all use cases. More specifically:				

	1. It will be composed of discrete modules addressing specific "competency questions", following best practices in ontology engineering- allowing these to be adopted standards or tightly managed development efforts with clear testability. 2. It needs to handle interoperability for different implementation aspects. 3. Meta-models, domain models, profiles and vocabularies need to be handled individually using appropriate specialised modelling mechanisms.
Relevant Pilot(s)	**ALL**
Relevant Task(s)	T2.1
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models Objective 2: Knowledge Exchange Mechanisms
Relevant Innovation(s)	1. Agriculture Interoperability Space 8. Unified Agriculture Ontology
Involved stakeholders/actors	Data modelers, component system designers, system architects, standardization organizations
Prerequisite(s)	
Type	Methodology
Priority Level	Mandatory
Identified by Partner(s)	OGCE, FhG.FIT
Comments/Remarks	DEMETER will require a complex set of interoperability agreements covering many different aspects of data exchange and semantic description. Common approaches tend to be one-dimensional – i.e. limited to a single model of interoperability – such as service interfaces, data schemas, openness and accessibility, etc. To achieve integration of component systems across the DEMETER project scope, it will be necessary to achieve interoperability across a range of such concerns. At this stage, it is not easy to identify a comprehensive model of interoperability that can be adopted, however by capturing each aspects where stakeholders need information about some aspect of a data exchange, and the role of various supporting infrastructures and components it will be possible to develop and exemplify a minimum necessary and sufficient interoperability architecture. The diversity of pilots and functional requirements in DEMETER demands and provides an

	opportunity to test innovation in this space.

Requirement ID	DK1.18	**Version**	0.2	**Last Update Date**	13/12/2019
Title	Simplified Profiles of Data Model				
Description	DEMETER needs to provide simple profiles of the general model suitable for individual pilot cases; these profiles will define “schemas” - or views, while the general model will define semantics - what objects can be identified and reused in different views.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Data modelers, implementers of systems, data integrators, data discovery agents, semantic technology experts, pilots’ data leads				
Prerequisite(s)	Comprehensive data model, profiling mechanism, identification of stakeholders’ requirements for simplified application specific subset of model.				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	OGCE				
Comments/Remarks	The DEMETER mode,l as suggested by the full range of requirements, will be a very large and complex, highly modular model using some powerful but very general standards. When implementing or using data for a specific component (via some API or data exchange) only a small subset of this model will be involved. The concept of “profile” is used to narrow general models down to specific simplified cases. The formalism of the profile descriptions allows simple views to be discovered and integrated safely into the common, complete, model as required. Profiles are akin to mapping (and may include mapping specifications) but they also allow for the simpler				

	and less semantically ambiguous case of just selecting a relevant subset of the common model for use in a particular Use Case.

Requirement ID	DK1.19	**Version**	0.2	**Last Update Date**	30/01/2020
Title	Semantic model that supports scalability and support of legacy systems				
Description	DEMETER needs to implement semantic interoperability in a scalable and sustainable way, e.g. by maintaining a dependency graph at the module level within each implementation rather than creating a temporary (project scoped) aggregated knowledge graph with no transparency of scope or provenance. It should support semantic interoperability for data originating from existing systems involved in the pilots (legacy systems). It should publish all domain-specific semantic interoperability resources in a canonical standards-based and interoperable fashion appropriate to the type of resource (e.g. vocabulary, schema, object model, profile, datatype, etc.)				
Relevant Pilot(s)	**All**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models				
Relevant Innovation(s)	8: Unified agriculture ontology				
Involved stakeholders/actors	Data publishers, system architects, infrastructure providers, standards organizations.				
Prerequisite(s)	Semantics publishing activities.				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	OGCE, ICCS				
Comments/Remarks	The complexity of the DEMETER project scope, and its interactions with related domains of knowledge, means that a significant number of components will be required to implement the range of pilot projects. Many of these components will be				

	information resources, such as ontologies, vocabularies, mappings and registers which will be used to drive data integration and processing functions. In order to keep these simple, the number will grow (there is a trade-off between a few very complex objects and a large number of simple objects). Experience has shown that small sets of complex objects are hard to maintain, but large sets of simpler objects require tools and infrastructure to manage dependencies. In the Java world, the Maven infrastructure manages dependencies across the typically hundreds of small libraries – but developers, once they embrace this, are freed from the curse of continually changing critical libraries and extreme complexity and risk in assessing what impacts changes may have. DEMETER should "plan to succeed" by assuming a sophisticated dependency management approach as a core architectural requirement.

Requirement ID	DK1.20	**Version**	0.2	**Last Update Date**	30/01/2020
Title	Governance Arrangements				
Description	DEMETER needs to specify governance arrangements for each component, determining who, when and how updates to the included components should be handled. This includes pragmatic project-scope governance of temporary resources, as well as requirements for governance of project resources that would ensure future interoperability.				
Relevant Pilot(s)	**All**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models				
Relevant Innovation(s)	9: Secure Agricultural data sharing services 8: Unified agriculture ontology				
Involved stakeholders/actors	Data publishers, system architects, infrastructure providers, standards organizations.				
Prerequisite(s)	Project architecture and persistence requirements.				
Type	Functional				

Priority Level	Mandatory
Identified by Partner(s)	OGCE
Comments/Remarks	To have a lasting impact and application beyond the initial pilots there needs to be a means to share semantic resources amongst a community of practice in a sustainable fashion. Resources need to meet FAIR principles (Findable, Accessible, Interoperable and Reusable). At the heart of "Reusable" is the issue of risk and trust – in order to reuse a resource a stakeholder needs to understand both the technical usefulness of the resource as well as how it may be "Accessible" in future. One aspect of this is "governance" – as exemplified by ISO 19135 (Procedures for Item Registration). Transparency of governance includes the policies and mechanisms by which resources may be created, reviewed and updated, and should also include statements around the persistence of identifiers and services.

Requirement ID	DK1.21	**Version**	0.2	**Last Update Date**	05/12/2019
Title	Abstract model for integrating sensors, processing and decision support systems.				
Description	DEMETER needs to have an abstract model for the general process of linking sensor data through processing chains into decision support systems, including how intermediate data products relate to sources and outputs. This can be based on an existing general model, or, if necessary, to create something new, to be pushed as an OGC and/or W3C general model spec.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models Objective 2: Knowledge Exchange Mechanisms				
Relevant Innovation(s)	5. Farm enabler dashboards 7. Cost- and power-effective IoT data acquisition 8. Unified agriculture ontology 14. Smart fruit pesticides management				

Involved stakeholders/actors	Data integrators, data discovery agents.
Prerequisite(s)	General interoperability model (DK1-17), Interoperability profiles for abstract components (sensors, processes etc.)
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	OGCE
Comments/Remarks	Many DEMETER pilots explicitly or implicitly involve aggregation of data into a decision support system (DSS). In general, it is assumed that DSS will have the ability to reuse available data, and such data may be used by many systems, including DSS. Usually, if not always, data from direct observation (including sensors) is processed in a pipeline to be delivered to the DSS in a form relevant to the decision criteria. Aggregation, interpolation, and signal (pattern) detection in time and space is usually performed to simplify rich observation data into summaries of state and trends of conditions of interest. In order to have reuse and interoperability of data and processing systems, the role of each in relation to both the DEMETER information model and the functional processing involved needs to be described. This will be relatively complex – and, if this is done on an ad-hoc basis, the effective complexity (ability to identify reusability) will grow exponentially with the number of examples. If simplified profiles of the data model are identified for a set of abstract processes (i.e. a profile for measurement of some environmental factor at farm scale at regular intervals over a year) - then the level of complexity grows more slowly, as much of the descriptive burden is handled by reusable patterns for common processes, and it will become possible to compare and reason over data descriptions based on similarity as well as specifics. Crudely, it should be possible to ask a data catalog what data sources are compatible with a given DSS input requirement. This will require a model of both such DSS data requirements and available processing steps that can be used to generate the data product from observational data. At the very least, capturing the DEMETER pilot scope will inform the development of a generalized approach for future interoperability standards.

6.2 Semantic mapping of AIM to dominant/standardised agrifood solutions

Requirement ID	DK2.1	**Version**	0.2	**Last Update Date**	24/01/2019

Title	Service wrappers and translators
Description	DEMETER needs to develop service wrappers and translators, also known as DEMETER providers and consumers, which will enable the different tools/platforms in a (regional/national) AKIS to expose and consume data in interoperable forms.
Relevant Pilot(s)	**ALL**
Relevant Task(s)	T2.2
Relevant Objective(s)	O2: Build knowledge exchange mechanisms O6: Demonstrate the impact of digital innovations across a variety of sectors and at European level
Relevant Innovation(s)	1. Agriculture Interoperability Space 11. Data integration across the entire dairy supply chain 15. Open AKIS for irrigated crops 17. Water Management Model and Coordination Broker
Involved stakeholders/actors	Solution providers, semantic technologies experts
Prerequisite(s)	AIM
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	PSNC, ICCS, OGCE, TECNALIA
Comments/Remarks	Such components include for instance, (a) components enabling to harness satellite data for applications in farm telemetry, with particular interest in crop monitoring and predictions, (b) components for crop monitoring and real-time analytics using real-time streaming data from wireless sensor networks, and (c) components with the capability to trigger alarms, notifications and/or recommendations in order to improve farm operations and productivity. The DEMETER provider-consumer services, deployed on the various components, translate and exchange data based on the AIM common data format with the utilization of lightweight data wrappers/translators. Hence, in order to develop these wrappers/translators each of the AKIS components should provide the specifications of the utilized data model-

	semantics and/or it should parse the returned content in AIM format. The translators will then implement mapping rules between the components' underlying data models and AIM to transform the data from the component to/from AIM ontology. This may also include syntactic and data conversion rules (e.g. mapping to common datum, timezones, etc).

Requirement ID	DK2.2	**Version**	0.2	**Last Update Date**	24/01/2019
Title	Mapping AIM with standard models				
Description	DEMETER should implement (semantic) mappings from standard and/or widely used ontologies/vocabularies with the AIM, enabling the semantic integration of data represented using any of these models. As part of the semantic mapping, DEMETER will need to identify logical connections between classes, properties, and objects across ontologies. The mappings will deal with cases in which, e.g. a class in one ontology is the intersection (or union) of two classes in another, or the complement of another class, or a simple object needs to be mapped to a complex class in another ontology, etc.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	O1: Analyse, adopt, enhance existing (and if necessary, introduce new) information models O2: Build knowledge exchange mechanisms				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Domain experts, Semantic technologies experts, data consumers				
Prerequisite(s)	AIM				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	PSNC, ICCS, OGCE				

Comments/Remarks	Data consumers benefit by being able to refer to standard models and terms in order to understand data content

Requirement ID	DK2.3	**Version**	0.2	**Last Update Date**	24/01/2019
Title	Semantic Interoperability				
Description	Support semantic interoperability, encompassing semantic integration (DK3). This will be realized through the implementation of the various DEMETER provider and consumer services (DK3.1), new ontologies and the mappings with existing ontologies/vocabularies (DK3.2), as well as the other mechanisms developed to facilitate data integration (see DK3).				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	Objective 1: Analyse, adopt, enhance information models.				
Relevant Innovation(s)	8.Unified agriculture ontology				
Involved stakeholders/actors	Consumers, Producers				
Prerequisite(s)	Existing ontologies/vocabularies, DK3.1, DK3.2, publishing mechanisms and standards for components required to publish in full.				
Type	Functional				
Priority Level	Mandatory				
Identified by Partner(s)	ICCS				
Comments/Remarks	This requirement implies full support for all semantic information through a complete data publishing, access, integration and use lifecycle (discovery, publishing, analysis, notification, etc. of the results of the integration are related but distinct requirements). This requirement focuses on whether sufficient information is				

	available to support integration and whether that information is accessible and interoperable.

Requirement ID	DK2.4	**Version**	0.2	**Last Update Date**	12/12/2019
Title	Mapping best practices				
Description	Follow best practices and approaches for generating the mapping between AIM and existing ontologies/vocabularies, including • Transformation of existing ontologies into common format, e.g., OWL, use of semantic rules or annotations/punning. • Reuse of AIM terms, and only extend it if necessary. In the latter case, reuse existing terms whenever possible, and only otherwise create new terms/extensions. • Use of appropriate mapping constructs/axioms, such as owl:equivalentClass and owl:equivalentProperty with OWL classes/properties; skos:closeMatch, skos:exactMatch, skos:broadMatch, skos:narrowMatch, and skos:relatedMatch with SKOS concepts; owl:sameAs for individuals, etc. • Treating of the mappings as “first class” components of a modular knowledge graph, making them available in line with FAIR principles, and governing them appropriately and transparently. • Consider mappings across different levels of specification granularity as well of abstractions using the appropriate mechanisms in a standardised way, e.g., mappings from meta-models to models (OWL subclassing); mappings between concepts at the same level of abstraction; mappings between controlled vocabulary terms; mappings between measurements and classifications (e.g. threshold values for "good" etc.); soft- vs. hard-typing mappings with classes with a sub-type property vs. specific sub-classes				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	O1: Analyse, adopt, enhance existing (and if necessary, introduce new) information models				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Domain experts, Semantic technologies experts				

Prerequisite(s)	AIM
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	ICCS, PSNC, OGCE, Tecnalia

Requirement ID	DK2.5	**Version**	0.2	**Last Update Date**	12/12/2019
Title	Tools for generating ontology mappings (semi-) automatically				
Description	Identify and select, if possible, suitable tools for the (semi-) automatic mapping of ontologies/vocabularies. Some example tools to be analysed include the Alignment API, PARIS and Map-On				
Relevant Pilot(s)	• Information model for water management: 1.1, 1.2, 1.3, 1.4, 3.1, 3.2 • Information model of crops, pests, treatment and fertilisation data: 1.3, 1.4, 2.2, 3.1, 3.2, 3.3, 5.1, 5.3 • Information model of soil data: 1.4, 3.2 • Information model for weather data: 1.4, 2.2, 3.1 • Information model of Vehicle data and emissions: 2.1 • Information model for farms and animals: 4.1, 4.2, 4.3, 4.4, 5.2, 5.3, 5.4 • Information model of farm economic data: 2.4 • Information model of status and field data: 1.3, 3.1, 3.4, 5.2 • Information model for the traceability of crops, dairy products, poultry products: 5.1, 5.2, 5.4				
Relevant Task(s)	T2.1				
Relevant Objective(s)	O1: Analyse, adopt, enhance existing (and if necessary, introduce new) information models				
Relevant Innovation(s)	8: Unified agriculture ontology				
Involved stakeholders/actors	Solution providers, Domain experts, Semantic technologies experts				
Prerequisite(s)	Data models should be based on existing ontologies				
Type	Functional				

Priority Level	Desirable
Identified by Partner(s)	PSNC, OGCE, ICCS, TECNALIA

Requirement ID	DK2.6	**Version**	0.2	**Last Update Date**	07/01/2020
Title	Identify tools to validate mappings				
Description	In order to facilitate the mapping between the Demeter AIM and existing ontologies, it is necessary to identify and select, if possible, suitable tools to validate the generated mappings. This is necessary because some of the mappings may be quite complex. For example, when a specific schema is mapped to a more general schema, then some schema elements may be replaced by use of a qualifying term in corresponding more abstract elements. In such cases, we need to validate the coverage of the mappings as well as the result of exercising a mapping against the target model. It would also be desirable to define a validation process and a simple reference implementation that can define test procedures to be integrated into traditional development tooling.				
Relevant Pilot(s)	**ALL**				
Relevant Task(s)	T2.1				
Relevant Objective(s)	O1: Analyse, adopt, enhance existing information models O2: Build knowledge exchange mechanisms				
Relevant Innovation(s)	8. Unified agriculture ontology				
Involved stakeholders/actors	Domain experts, Semantic technologies experts				
Prerequisite(s)	Standardised approaches to publishing schema, terms, mappings between schema, mappings between terms and binding of term ranges to schema (profiles of schema)				
Type	Functional				

Priority Level	Mandatory
Identified by Partner(s)	PSNC, ICCS, OGCE
Comments/Remarks	Different tools may be required for different aspects. Mappings may be trivial - can be limited to schema mappings or term mappings - but many may be more complex - for example when a specific schema is mapped to a more general schema (the usual case) then some schema elements may be replaced by use of a qualifying term in corresponding more abstract elements: <pig>123</pig> => <animal><type>swine</type><id>123</id></animal> Tools such as VocBench may support effective means of validating term mappings, and schema mapping tools may be available that can be adapted. Tools should be evaluated on multiple criteria: 1) which parts of mappings they can validate 2) effectiveness at validation 3) ease of integration into testing mechanisms 4) accessibility to relevant stakeholders 5) flexibility 6) overhead of familiarisation with tool-specific UI and data management paradigms In general, however, a validation process needs to be defined and a simple reference implementation that can define test procedures needs to be integrated into traditional development tooling - i.e. wrapped up as a test case with test case data samples and executed using readily available orchestration tools. A ubiquitous language like python and tools like pySHACL can be used to validate input and output shapes using available standard constraints languages. A regression testing using some form of continuous integration will be required to ensure that evolving quality and scope of mappings for more complex cases continue to work reliably for the simpler cases that will probably be validated and deployed first.

Requirement ID	DK2.7	**Version**	0.1	**Last Update Date**	04/12/2019
Title	Select relevant existing ontologies to align with AIM				

Description	Identify and select relevant standards and/or widely used ontologies/vocabularies to align with the AIM and identify the key terms in each of them that would need to be aligned. Some examples, classified by the type of data, include:
	• Water management:
	  ○ Saref4agri → s4agri:WateringSystem
	  ○ INSPIRE → WaterManagement, irrigation
	• Crops and pests:
	  ○ FOODIE (INSPIRE based) → cropType
	  ○ FIWARE → AgriCrop, AgriPest
	  ○ rmAgro
	  ○ drmCrop
	  ○ AGROVOC → crops, plant products
	• Soil data and other sensor measurements:
	  ○ Saref4agri → s4agri:SoilMoisture, s4agri:SoilTemperature
	  ○ AFarCloud → observations (IoT devices)
	  ○ Soilphysics → soilProperty
	  ○ SOSA/SSN → sosa:Observation
	• Weather data:
	  ○ Agrifood Data Models → Weather Observed, Weather Forecast, Weather Alert
	  ○ Saref4agri → s4agri:AmbientHumidity, s4agri:AirTemperature
	• Vehicle data:
	  ○ FOODIE → Transport data model
	  ○ AFarcloud → hierarchy of robotic vehicles (UGVs, AUvs)
	• Farm data:
	  ○ FOODIE → Holding, Site, Plot, ManagementZone, ProductionType, CropSpecies, etc.
	  ○ AgriFarm
	  ○ Saref4agri → s4agri:Farm
	• Field data
	  ○ FOODIE → cropSpecies
	  ○ FIWARE → AgriCrop
	  ○ Saref4agri → s4agri:Crop, s4agri:PlantGrowthStage;
	• Animals and dairy farms
	  ○ Agrifood → Animal
	  ○ Saref4agri → s4agri:Animal, s4agri:MilkingSensor, s4agri:ActivitySensor
	  ○ INSPIRE → Animals and animals health

	○ AFarCloud → Dairy farms • Treatments ○ FOODIE → Treatment, TreatmentPlan
Relevant Pilot(s)	• Information model for water management: 1.1, 1.2, 1.3, 1.4, 3.1, 3.2 • Information model of crops, pests, treatment and fertilisation data: 1.3, 1.4, 2.2, 3.1, 3.2, 3.3, 5.1, 5.3 • Information model of soil data: 1.4, 3.2 • Information model for weather data: 1.4, 2.2, 3.1 • Information model of Vehicle data and emissions: 2.1 • Information model for farms and animals: 4.1, 4.2, 4.3, 4.4, 5.2, 5.3, 5.4 • Information model of farm economic data: 2.4 • Information model of status and field data: 1.3, 3.1, 3.4, 5.2 • Information model for the traceability of crops, dairy products, poultry products: 5.1, 5.2, 5.4
Relevant Task(s)	**T2.1**
Relevant Objective(s)	Objective 1: Information Modelling
Relevant Innovation(s)	3. Agricultural automation and control 8: Unified agriculture ontology 11: Data integration across the entire dairy supply chain 17: Water Management Model and Coordination Broker
Involved stakeholders/actors	Solution providers
Prerequisite(s)	Existing relevant ontologies
Type	Functional
Priority Level	Mandatory
Identified by Partner(s)	PSNC, OGCE, ICCS, TECNALIA

7 Initial design of the Agricultural Information Model (AIM)

In line with best practices and recommendations, the specification of DEMETER AIM will follow a modular approach in a layered architecture, enabling among others:

1. eased interoperability with existing models by reusing available (well-scoped) models in the modules, instead of defining new terms, whenever possible,
2. easy mapping/alignment with other models, by module instead of the whole model,
3. easy extension of the domain/areas covered in AIM with additional modules,
4. easy extension of the domain model, by modifying only specific modules,
5. easy mapping to top-level/cross-domain ontologies.

More specifically, Section 7.1 presents the AIM core metamodel, which follows the NGSI-LD meta-modeling approach; Section 7.2 presents the cross-domain ontology used, i.e. the set of generic models which aim at providing common definitions for all agrifood domain handled by AIM and at avoiding conflicting or redundant definitions of the same classes at the domain-specific layer; Section 7.3 presents the domain specific ontologies developed for the AIM which model information such as crops, animals, agricultural products as well as farms and farmers just to mention some of the most important concepts included in these ontologies; and, Section 7.4 describes the metadata schema used by AIM and which expresses semantics related to meta-information about the datasets based on the cross-domain and domain specific ontologies previously presented.

7.1 Core meta-model

A meta-model, as its names implies, is a model of a model. Meta-models are typically used for different purposes. For instance, they can be used for the specification of modelling language constructs in a standardized, platform independent manner [HaPa09], to specify and restrict a domain in a data model and systems specification [IvVo11], or to provide an explicit model of the constructs and rules needed to build specific models within a domain of interest [Wel]. In fact, as noted in [Wel], meta-models can be viewed from three different perspectives: i) as a set of building blocks and rules used to build models; ii) as a model of a domain of interest; iii) as an instance of another model. In the context of the DEMETER meta-model, we are considering it as the first perspective.

Related to our context, it is important to highlight the relation between meta-models and ontologies. Ontologies provide shared vocabularies formally describing entities, such as concepts, properties and relations along with logical assertions (statements, rules), of a particular domain or that are common across multiple domains. They can be defined at different abstraction levels: top-level (foundation), domain, application. For instance, top-level ontologies define entities common across all/multiple domains and are the basis to support semantic interoperability among different domain-specific ontologies. Meta-models, on the other hand, are the explicit specification (constructs and rules) of how to build domain-specific models. They are targeted mainly at a

structural specification of models, i.e. they are not intended to fully define their (logical) semantics [HaPa09]. Nevertheless, as stated in [Wel], a valid meta-model is an ontology, but ontologies may be defined at different abstraction levels, i.e., the meta-model may be regarded as an ontology used by modelers and integrated with ontologies at different abstraction levels. Thus, a foundational ontology (at the same abstraction level as a meta-model) and a domain ontology (at the same abstraction level as a (design) model) [HeSe11] can be integrated via an appropriate semantic mapping, which may be yet another ontology carrying a complementary set of semantics.

Based on the previous discussion of the use and benefits of meta-models, our task was to decide our meta-modeling approach. After analysing different models and approaches (e.g., [HaPa09], [HeSe11], [SaKa07], [OMV14], [PaLi10]), we decided to follow the NGSI-LD meta-modeling approach [NGS1]. This approach is already a standard of the European Telecommunications Standards Institute (ETSI), whose mission is to make it easier for end-users, city databases, IoT and 3rd party applications to exchange information. Moreover, NGSI-LD is an evolution of the NGSI context interface family, particularly the FIWARE NGSI v2 information model, which was evolved by ETSI ISG CIM initiative to support linked data, property graphs and semantics, which is also a priority in DEMETER. Those standards were focused on the management of context information, which facilitate the development of smart solutions for different domains such as smart cities, smart industry, smart agrifood, and smart energy[42]. Here, context comprises all characteristics of all the entities (physical and nonphysical) involved in a target system/environment, as well as their states and other dynamic properties, together with relationships that stand for actual and virtual connections between them [ETS6]. Furthermore, the NGSI-LD meta-model provides a formal basis for representing "property graphs" using RDF/RDFS/OWL, making it possible to perform back and forth conversion between datasets based on the property graph model and linked data datasets that rely on RDF using blank-nodes reification. Moreover, the use of JSON-LD allows also the semantic referencing, where elements in the model can be matched to entities in well-known and/or standard ontologies. Following the NGSI-LD meta-modeling approach does not imply that we need to use either domain models such as FIWARE nor the domain model management methodology and tooling, however we can learn from, adopt, adapt and improve these aspects as necessary.

The main components of a meta-model can include:

1. Semantic modelling language (OWL) – for domain models as classes and properties
2. Terminology definition language (SKOS) – concept labels and definitions
3. Structural (schema/frame) based (SHACL and RDF-Datacube) – data shapes
4. Binding these elements together (profiles)

Most domain models are in fact profiles of more general models, and are often defined informally through packaging frames or definitions into specific artefacts, where the presence of a copy of a definition in a file

[42] https://www.fiware.org/developers/

indicates its acceptance within a domain model, even though no axioms are present to state this intention. Such models are difficult to integrate, as it becomes necessary to track the source of each definition using a concept (the file) that itself has no semantic value. A formalism for profiles, using the elements of the meta-model can be used to both capture the intent of how domain models interact beyond just providing a list of properties (are they supposed to be compatible)?Profiles based on SHACL and RDF-Datacube, and described using the Profiles Vocabulary [PROF19] provide a general metamodel for integrating these different functional meta-model components, along with implementation resources such as JSON-LD context definitions.

7.1.1 NGSI-LD considerations

The NGSI-LD Information Model is defined at three levels. At the higher level, there are the foundation classes which correspond to the Core Meta-model and the Cross-Domain Ontology (see Figure 31 below). The former concerns the formal specification of the "property graph" model [Ro15]. The latter includes a set of generic, transversal classes which are aimed at avoiding conflicting or redundant definitions of the same classes in each of the domain-specific ontologies. Below these two levels, domain specific ontologies or vocabularies are devised.

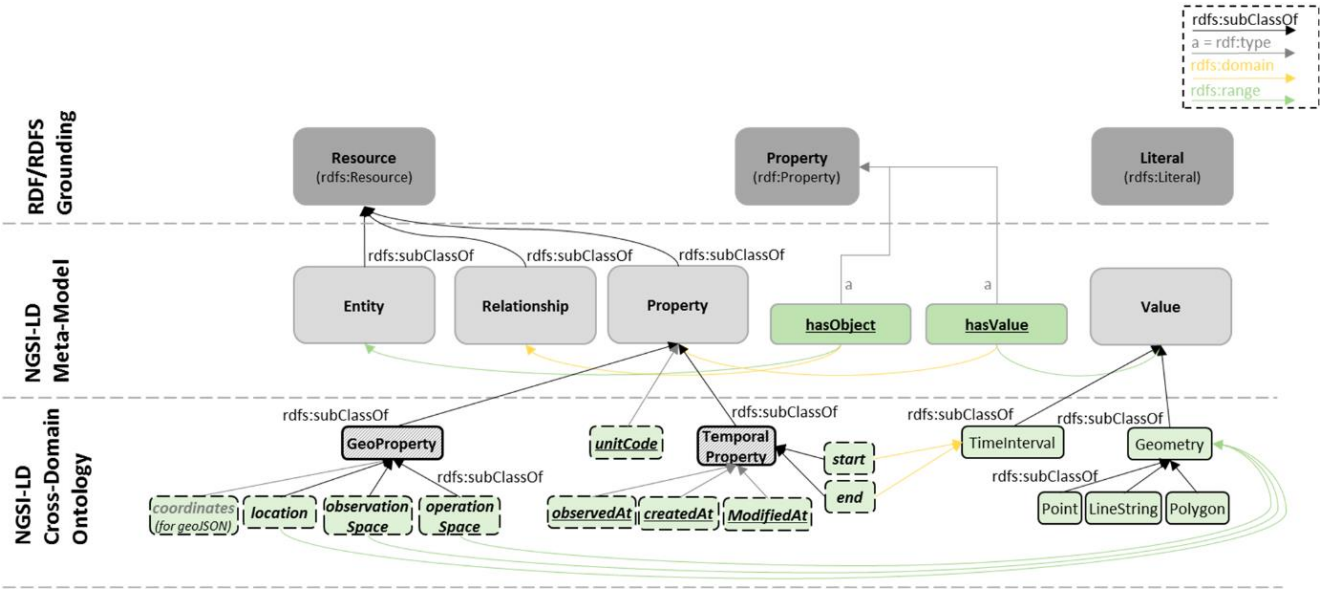


Figure 31. Meta-model and Cross-Domain Ontology High-Level View

NGSI-LD uses JSON-LD as main serialisation format, which provides the key advantage that terms can be defined in a separate document, referenced by an @context statement. In particular, the @context in JSON-LD is used to map terms provided as strings to concepts specified as URIs (ideally in ontologies).

NGSI-LD adopts a graph-based meta-model solution along with blank node reification, which is “especially convenient when the graph is serialized with JSON-LD because blank nodes do not explicitly appear in the textual

serialized description, and actually show up only when it is represented as an RDF graph. It is thus possible for a developer to generate the JSON-LD payload of an API in a form that is very similar to what he would have generated in plain JSON [ETS6].

7.1.2 Separation of semantic referencing and structural descriptions

According to the specification [ETS6], NGSI-LD information model separates semantic referencing (as used in the Semantic Web) from the actual structural description. The structural description may be decomposed into a base structural graph whose nodes are physically matched entities, and an overlay layer that captures the way in which these entities are clustered into subgraphs.

The semantic referencing in NGSI-LD is in theory based on standard RDF/RDS/OWL typing and public ontologies. Accordingly, all nodes and edges of the structural graph are matched to several relevant classes/categories of such ontologies, which together characterize the features shared by all the instances of these classes.

The structural graph is as a model of the structural description of an environment and captures the relationships between the different subsystems that make up this environment. This is, according to the specification, to some extent independent of the overlaying semantic referencing and it could be considered to "stand on its own", even without such referencing.

In contrast to the semantics “per resource” that RDF is meant to describe (e.g., via referencing), the structural graph has a different kind of semantics of its own that apply to the graph as a whole, such as e.g. when a graph captures and matches the structure of a physical network like a power grid or a water distribution network.

In the implementation, though, the semantic referencing mentioned here is not really followed, as discussed later in the insights.

7.1.3 NGSI-LD meta-model

According to the specification [ETS6], the NGSI-LD meta-model provides a formal basis for representing "property graphs" using RDF/RDFS/OWL. This makes it possible to perform back and forth conversion between datasets based on the property graph model and linked data datasets that declare more formal semantics using RDF. This could be described as raising the semantic expressivity of RDF triples to the level of property graphs, as for instance, property graphs may use predicates as subjects of other predicates (properties of properties and properties of relationships). Conversely, it may be described as grounding the semantics of property graph elements in discoverable definitions and using this to constrain arbitrary and non-interoperable proliferation of similar property graph patterns for the many specific cases that need to be modelled.

A graph-based model was chosen as it enables to capture the complex structure and inter-entity relationships describing the characteristics of entities (physical and non-physical) involved in systems for smart solutions,

which make up the context information. Such information may be natively structural information only, with no semantic definitions from the beginning; *the semantics of this* context *may be added in a later stage of graph enrichment.* This is the case of the current implementation of the NGSI-LD context, where semantics are not defined yet (see discussion below).

7.1.4 Context documents and Schemas

7.1.4.1 Roles

NGSI-LD uses both JSON Schema and JSON-LD Context documents for their respective roles: Schema documents describe which elements must be present and Context documents provide information about what they mean (mainly by providing a Web URI to uniquely identify things). It is not defined what information those Web addresses provide. But it is necessary to know and provide semantic descriptions of elements to successfully integrate data in DEMETER.

Content (property values) needs to be explained and controlled too - this leads to challenges factoring out standardised content from semantic data models.

Furthermore, NGSI-LD uses an object reference scheme based on its internal meta-model that needs tool support to turn object references into addresses for those objects in the NGSI environment, which is not natively able to support Linked Data style object references. This seems to be an idiosyncrasy to support backwards compatibility between NGSI-LD and NGSI.

Tools exist[43] to derive NGSI-LD context documents from NGSI data schemas, such as those described below for FIWARE implementation of NGSI-LD, but they are minimally documented and may have limited generality. NGSI data models are just schemas, rather than semantic models, so these context documents do not provide the basis for semantic definition or description, and hence support any processing of such data. What they do provide is a canonical way to link the schemas to such definitions.

Given that DEMETER is not just an innovation project, but rather a “large-scale deployment of farmer-driven, interoperable smart farming IoT-based platforms”, it is benefitial to follow such a flexible process that will not slow down the use cases. Even though that it is good in theory in order to enforce strict mechanisms ensuring data consistency, in practice it may be difficult to implement such an approach in large-scale deployments in operational environments. Each Use Case should have some level of flexibility and be encouraged to contribute to and share their living data models. However, there is a demonstrable practicality of the approach of developing tools to support some degree of consistency with a highly modular approach. It is recommended to proceed by developing or adapting a similar suite of tools to automate validation and documentation of complex

[43] https://github.com/FIWARE/data-models/tree/master/tools

specifications required for DEMETER.

7.1.4.2 The NGSI-LD @context

The core NGSI-LD (JSON-LD) @context[44] is defined as a JSON-LD @context which contains:

- The core terms needed to represent the key concepts defined by the NGSI-LD Information Model, including the meta-model and cross-ontology.
- The terms needed to represent all the members that define the API-related Data Types.

The challenges inherent in the NGSI-LD context include:

- URIs in the context (e.g., https://uri.etsi.org/ngsi-ld/location or https://uri.etsi.org/ngsi-ld/Time) do not resolve to either human or machine-readable descriptions of what properties mean. (They should do both via content negotiation);
- ad-hoc approach definitions, ignoring existing domain standards - particularly spatio-temporal properties not aligned with domain standards (OGC and W3C, which maintain a formal liaison to develop best practices). For example:

```
unitCode: "https://uri.etsi.org/ngsi-ld/unitCode",
location: "https://uri.etsi.org/ngsi-ld/location",
observationSpace: "https://uri.etsi.org/ngsi-ld/observationSpace",
operationSpace: "https://uri.etsi.org/ngsi-ld/operationSpace",
GeoProperty: "https://uri.etsi.org/ngsi-ld/GeoProperty",
TemporalProperty: "https://uri.etsi.org/ngsi-ld/TemporalProperty",
timeInterval: "https://uri.etsi.org/ngsi-ld/timeInterval",
geoQ: "https://uri.etsi.org/ngsi-ld/geoQ",
temporalQ: "https://uri.etsi.org/ngsi-ld/temporalQ",
geometry: "https://uri.etsi.org/ngsi-ld/geometry",
coordinates: "https://uri.etsi.org/ngsi-ld/coordinates",
georel: "https://uri.etsi.org/ngsi-ld/georel",
geoproperty: "https://uri.etsi.org/ngsi-ld/geoproperty”
```

7.1.4.3 FIWARE contexts

The FIWARE contexts are evolving from a very large flat list to a repository of data models[45]. Some common

[44] https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld
[45] https://github.com/smart-data-models

models are factored out for reuse[46]. This avoids some of the limitations of the NGSI models, but introduces a duplicate model covering the same ground - FIWARE may be seen as a discrete interoperability domain within an NGSI world.

The FIWARE approach provides a translation between "normalised" models and simple schemas. For example, the following object:

```
"id": "urn:ngsi-ld:AgriApp:72d9fb43-53f8-4ec8-a33c-fa931360259a",
"type": "AgriApp",
"dateCreated": "2017-01-01T01:20:00Z",
"dateModified": "2017-05-04T12:30:00Z",
"name": "Wine track"
```

can be normalised into a json-ld object[47]:

```
"@context": { ["https://schema.lab.fiware.org/ld/context",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"]
      "id": "urn:ngsi-ld:AgriApp:72d9fb43-53f8-4ec8-a33c-fa931360259a",
      "type": "AgriApp",
       "createdAt": "2017-01-01T01:20:00Z",
       "modifiedAt": "2017-01-04T12:30:00Z",
      "name": {
        "type": "Property",
        "value": "Wine track"
      }
}
```

Schemas are defined for the simple model; it is still somewhat unclear whether the simple schema is operationally used.

Most importantly, FIWARE is referencing an external standard for geo concepts (GeoJSON) rather than re-inventing the wheel; however, it is only doing this for data types in JSON schema, without declaring the equivalent semantics in a reusable context document.

FIWARE tooling aggregates the complete set of data models into a single extensive context document. This represents a promising but not fully scalable start, and its roadmap indicates it is still very immature and

[46] https://github.com/smart-data-models/data-models/blob/master/common-schema.json
[47] https://github.com/smart-data-models/dataModel.Agrifood/blob/9eb5e73f13ffbfa646ba132b1bf62ad1c2c53b31/AgriApp/example-normalized-ld.jsonld

untested.

The challenges of the FIWARE approach include:

- This reference context is a collation of every term used in every data model in the FIWARE repository – consequently, it is both large and would be likely to be changing regularly, so it could not be reliably cached, unless a system was not intended to be extensible to handle new data models.
- It is unclear if and how clashes between data models reusing terms for different purposes or using different URI references for the same term would be handled.
- **Very limited re-use of existing ontologies/vocabularies via semantic referencing, thereby mostly defining ad-hoc terms, and without explicit semantics.**

7.1.4.4 Profiles

FIWARE data model represents a *profile* of NGSI-LD API, but not an interoperable profile of the NGSI-LD core data model. Each FIWARE data model conforms to FIWARE rules including core element reuse and structure of the specification artefact sets - i.e. is a profile of the FIWARE profile of NGSI-LD API.

This represents an interoperability contract that is not explicitly stated, but can be inferred from re-use of contexts and commonality across schemas.

In order to address such issues, DEMETER will define data models that either conform to an equivalent DEMETER profile of NGSI-LD, or relevant FIWARE models, depending on the suitability of those models.

Whenever some common data structure is needed in different contexts, this can be factored out into a separate profile and shared. Tooling to handle consolidating the inheritance chain of profiles can be simply implemented and could be based on an augmentation of the FIWARE tools if required.

As a result, an example might look like:

```
"@context": { ["http://example.org/profiles/demeter/AnimalTracking",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
      "id": "urn:ngsi-ld:demeter:IndoorAnimalTracking"
}

"@context": { ["https://schema.lab.fiware.org/ld/context/Animal",
        "https://schema.lab.fiware.org/ld/context/Location",
        "http://example.org/profiles/demeter/MovingObject",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
      "id": "urn:ngsi-ld:demeter:AnimalTracking"
}
```

where the "AnimalTracking" profile inherits from reusable models for Animals, Location and MovingObjects.

7.1.5 NGSI-LD summary

NGSI-LD approach is well founded, following a layered architecture and based on the increasingly popular JSON-LD serialisation format. Conceptually, it enables the good sides of two “worlds”: the benefits of linked data and underlying RDF-based reasoning tools and querying (enabling data integration, knowledge discovery, etc.), and the richer expressivity of property graphs (using predicates as subjects of other predicates).

The current challenges we foresee, which could also be used as feedback for future developments in that community, are more on the implementation of this approach:

- The current NGSI-LD context is a simple flat schema that includes the meta-model and cross ontology terms without any explicit semantics. Except from some property JSON types (@type: DateTime, id), there are no definition that a term is a class, a property with explicit information about the type of property (e.g., relation, datatype), constraints on domains/ranges, cardinality, taxonomic relations, or other axioms. Of course, the JSON @type would allow to infer that a given term is a relation (@type: @id), but even those with @type: DateTime are not defined explicitly with the type of property it is, as DateTime (https://uri.etsi.org/ngsi-ld/DateTime) is not having any explicit semantic information.
- The terms are not mapped to any standard and/or well-known ontologies/vocabularies (no reuse)**.** NGSI-LD specification discusses such approach via the semantic referencing, but the context implementation is not including them; perhaps they are considered to *be added in a later stage* (as also mentioned in the documentation). There is also available documentation (see Annex B of [ETS6]) discussing mappings to some well-known ontologies/vocabularies (such as oneM2M, W3C WoT Thing Description, W3C Time Ontology and SAREF); however, no implementation seems to be available to allow any integration. In fact, it is not clear, how such mappings would be implemented from the documentation reviewed.
- Other modules/profiles (domain vocabularies) are defined in the same way, i.e., simple flat schemas with no mapping/reuse of existing standards and/or well-known ontologies. For instance, FIWARE Data Models @context[48] is used in many of the provided examples and is part of the full @context[49] (which also includes the core @context) of NGSI-LD. FIWARE @context defines many entities related to different FIWARE related domains. The full list set of models, called the FIWARE Smart Data Models[50], provide different json schemas and data examples in json and json-ld. The project GSMA IoT also provides a repository of different NGSI-LD entities from different domains[51], more complete than FIWARE, with a

[48] https://fiware.github.io/data-models/context.jsonld
[49] https://fiware.github.io/data-models/full-context.jsonld
[50] https://github.com/smart-data-models/data-models
[51] https://github.com/GSMADeveloper/NGSI-LD-Entities

specification, json-ld context example and data example in json-ld. An important difference, though, is that GSMA IoT includes some references to well-known ontologies or vocabularies. However, such references are very few and the context are only samples. Most of the terms are still defined ad-hoc.

- The flat schema implementation approach is not scalable, and difficult to maintain.
- The only semantic information available is in fact included in the encoding of data itself, and it is provided by the meta-model (e.g., an element is a property or a relationship). For instance, the encoding of a FIWARE agri-parcel entity is (partially) below (the full encoding of the example is also available for download[52]).
- There is sufficient complexity and evidence for the benefits for adaptation of tools to manage, collate, validate and document the DEMETER AIM using a similar approach to FIWARE, but with extended capabilities as required: for example, to create and exploit more interoperable intermediate profiles.

```
"@context": { ["https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
        "https://fiware.github.io/data-models/context.jsonld"]
      "id": "urn:ngsi-ld:AgriParcelRecord:8f5445e6-f49b-496e-833b-e65fc97fcab7",
      "type": "AgriParcelRecord",
      "createdAt": "2017-01-01T01:20:00Z",
      "modifiedAt": "2017-05-04T12:30:00Z",
      "source": "https://source.example.com",
      "dataProvider": "https://provide.example.com",
      "entityVersion": "2.0",
      "hasAgriParcel": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:AgriParcel:d3676010-d815-468c-9e01-25739c5a25ed"
      }
      "soilTemperature": {
        "type": "Property",
        "value": 27,
        "unitCode": "CEL",
        "observedAt": "2017-05-04T12:30:00Z"
      }
      "observedAt": {
        "type": "Property",
        "value": "2017-05-04T10:18:16Z"
      }
}
```

[52] https://github.com/rapw3k/DEMETER/blob/master/ngsi-ld-fiware-parcel-example.jsonld

The transformation of that json-ld into RDF would be as follows:

```
@prefix ns0: <https://uri.etsi.org/ngsi-ld/> .
@prefix ns1: <https://uri.etsi.org/ngsi-ld/default-context/> .
@prefix ns2: <https://uri.fiware.org/ns/data-models#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<urn:ngsi-ld:AgriParcelRecord:8f5445e6-f49b-496e-833b-e65fc97fcab7>
  a <https://uri.fiware.org/ns/data-models#AgriParcelRecord> ;
  ns0:createdAt "2017-01-01T01:20:00Z"^^ns0:DateTime ;
  ns1:entityVersion 2 ;
  ns0:modifiedAt "2017-05-04T12:30:00Z"^^ns0:DateTime ;
  ns2:dataProvider "https://provider.example.com"^^xsd:string ;
  ns2:hasAgriParcel [
    a ns0:Relationship ;
    ns0:hasObject <urn:ngsi-ld:AgriParcel:d3676010-d815-468c-9e01-25739c5a25ed>
  ] ;
  ns2:observedAt [
    a ns0:Property ;
    ns2:value "2017-05-04T10:18:16Z"^^xsd:string
  ] ;
  ns2:soilTemperature [
    a ns0:Property ;
    ns0:unitCode "CEL"^^xsd:string ;
    ns2:observedAt "2017-05-04T12:30:00Z"^^ns0:DateTime ;
    ns2:value 27
  ] ;
  ns2:source "https://source.example.com"^^xsd:string .
```

As the FIWARE @context does not link to any ontology, but the entities are defined ad-hoc, there are no explicit semantics. As a result, many advantages of the linked data and underlying RDF-based reasoning tools and querying cannot be easily or directly exploited, e.g., (automatic) data link discovery (integration), (automatic) model alignment for data integration, validation of conformance of data with the model with a simple reasoner, inferencing on the data to discover new knowledge, specialisations (taxonomy) with inheritance of axioms.

7.1.6 DEMETER AIM considerations

Our approach for the design of DEMETER AIM, discussed at the beginning of this section, is similar and in line with the NGSI-LD approach, i.e., modular in a layered architecture. Our first design choice, though, was to decide

whether to follow a 2-layer approach (top-level/cross domain + domain ontologies) with direct grounding on RDF/RDFS/OWL or a 3-layer approach as in NGSI-LD that includes the property graph meta-model layer (grounded on RDF/RDFS). After further analysis of the NGSI-LD specification, we decided on the latter for the following reasons:

1. It allows DEMETER AIM to be compliant and easily integrated with NGSI-LD data and models, thus facilitating the integration with existing datasets based on these models that may be relevant to DEMETER.
2. It allows natively the representation of the rich and complex context information of different entities (e.g., systems/platforms/environments) typical within IoT (or WoT) applications, where the context includes the set of properties characterizing these entities, together with the set of relationships that enmesh them together, and the properties of these relationships and properties. This was the main motivation of NGSI-LD and it is also a very important aspect for DEMETER.
3. It allows to have the best of two "worlds": property-graphs and linked data. It allows to perform back and forth conversion between datasets based on the property graph and linked data datasets that rely on the RDF framework. As described in [ETS6], property graphs are the implicit semi-formal data models underlying most present-day graph databases, which have gained widespread use especially in the industry (as opposed to academia). They make it possible to attach properties (defined as key-value pairs) to relationships and other properties, a feature which RDF does not directly support, but they lack the standardization and formal underpinnings of RDF and do not interoperate directly with linked data and other RDF datasets. Also, they do not lend themselves to reasoning with RDF-based reasoning tools or querying with standard query languages such as SPARQL.

Thus, DEMETER AIM follows the same 3-layer architecture of NGSI-LD, including a property graph meta-model layer (grounded in RDF/RDFS), a cross-domain ontologies layer, and the domain/application ontologies. However, as opposed to NGSI-LD, DEMETER AIM will implement the cross-domain and domain/application layers by reusing existing standards and/or well-known ontologies/vocabularies as much as possible from the outset, thereby implementing semantic referencing. As an example, consider the following agriculture management zone using FOODIE ontology[53] as the underlying model encoded in RDF/turtle.

```
@prefix ns0: <http://foodie-cloud.com/model/foodie#> .
@prefix ns1: <http://www.w3.org/2001/XMLSchema#> .
@prefix ns2: <http://www.opengis.net/ont/geosparql#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://w3id.org/foodie/core/ManagementZone/4>
  a <http://foodie-cloud.com/model/foodie#ManagementZone> ;
```

[53] http://agroportal.lirmm.fr/ontologies/FOODIE

```
  ns0:code "CODA4"^^xsd:string ;
  ns1:entityVersion 2 ;

  ns0:creationDateTime "2015-12-01T00:00:00"^^xsd:dateTime ;
  ns0:cropSpecies <http://w3id.org/foodie/core/CropType/20> ;
  ns0:holdingZone <http://w3id.org/foodie/core/Plot/1> ;
  ns0:originType <http://w3id.org/foodie/core/OriginTypeValue/1> ;
  ns0:zoneAlert <http://w3id.org/foodie/core/Alert/4> ;
  ns1:hasGeometry <http://w3id.org/foodie/core/ManagementZone/4/geometry> ;
  rdfs:label "ManagementZone #4"^^xsd:string .
```

With DEMETER AIM, we would define an agriculture model module/profile as a JSON-LD @context, which defines the terms used in DEMETER by reusing existing standards and/or well-known ontologies/vocabularies, such as Saref4Agri or FOODIE, i.e., mapping DEMETER terms to the reused ontology/vocabulary terms. A partial example of such @context[54] (using FOODIE ontology for demonstration purposes) would be as follows.

```
"@context": {
        "xsd" : "http://www.w3.org/2001/XMLSchema#",
        "Nutrients": "http://foodie-cloud.com/model/foodie#ProductNutrients",
        "Plot": "http://foodie-cloud.com/model/foodie#Plot",
        "DoseUnit": "http://foodie-cloud.com/model/foodie#DoseUnit",
        "TreatmentPlan": "http://foodie-cloud.com/model/foodie#TreatmentPlan",
        "ManagementZone": "http://foodie-cloud.com/model/foodie#ManagementZone",
        "Intervention" : "http://foodie-cloud.com/model/foodie#Intervention",
        "CropSpecies" : "http://foodie-cloud.com/model/foodie#CropSpecies",
        "Treatment" : "http://foodie-cloud.com/model/foodie#Treatment",
        "Holding" : "http://inspire.ec.europa.eu/schemas/af/3.0#Holding",
        "code" : "http://foodie-cloud.com/model/foodie#code",
        "creationDateTime" : {
                        "@id" : "http://foodie-
cloud.com/model/foodie#creationDateTime",
                        "@type": "xsd:dateTime"
        },
        "cropSpecies" : {
                        "@id" : "http://foodie-cloud.com/model/foodie#cropSpecies",
                        "@type": "@id"
        },
```

[54] https://github.com/rapw3k/DEMETER/blob/master/DEMETER-agricontext.jsonld

```
        "originType" : {
                        "@id" : "http://foodie-cloud.com/model/foodie#originType",
                        "@type": "@id"
        },
        "zoneAlert" : {
                        "@id" : "http://foodie-cloud.com/model/foodie#zoneAlert",
                        "@type": "@id"
        },
        "holdingZone" : {
                        "@id" : "http://foodie-cloud.com/model/foodie#holdingZone",
                        "@type": "@id"
        }
}
```

Then, the encoding of the same management zone presented above in JSON-LD using DEMETER AIM would look like the listing below (also available for download[55]), which could be easily transformed back to RDF[56] to get the same listing as above. Note that in addition to the agriculture context, we are adding two more terms to the context in this example (namely: label and geometry); however, such terms would be defined in the future in different profiles/modules at the cross-domain level (e.g., geospatial model)

```
"@context": {
        ["https://rapw3k.github.io/DEMETER/DEMETER-agricontext.jsonld",
         {"label" : "http://www.w3.org/2000/01/rdf-schema#label",
           "geometry": {
              "@id": "http://www.opengis.net/ont/geosparql#hasGeometry",
              "@type":"@id"
             }
         }
    ],
        "@id" : "http://w3id.org/foodie/core/ManagementZone/4",
        "@type": "ManagementZone",
        "label": "ManagementZone #4",
        "code": "COD4",
        "creationDateTime" : "2015-12-01T00:00:00",
        "cropSpecies" : "http://w3id.org/foodie/core/Croptype/20",
        "holdingZone" : "http://w3id.org/foodie/core/Plot/1",
        "originType" : "http://w3id.org/foodie/core/OriginTypeValue/1",
```

[55] https://github.com/rapw3k/DEMETER/blob/master/managementZone4-example.jsonld
[56] http://www.easyrdf.org/converter

```
        "zoneAlert" : "http://w3id.org/foodie/core/Alert/4",
        "geometry" : "http://w3id.org/foodie/core/ManagementZone/4/geometry"
}
```

Note, however, that if we would like to use the expressivity of the property graph model (to raise the semantic expressivity of RDF triples to the level of property graphs), we would first define our core meta-model @context[57] (same as for NGSI-LD) as the listing below:

```
"@context": {
      "id": "@id",
      "type": "@type",
       "value": "https://uri.etsi.org/ngsi-ld/hasValue",
      "object": {
        "@id": "https://uri.etsi.org/ngsi-ld/hasObject",
        "@type": "@id"
      },
      "Property": "https://uri.etsi.org/ngsi-ld/Property",
      "Relationship": "https://uri.etsi.org/ngsi-ld/Relationship"
}
```

Then we would be able to attach properties to relationships or other properties (i.e., using the property graph model). So, in our previous example, if we would like to attach properties to one of our data type properties (e.g., code to include for instance the codelist name or organisation name giving such code), and to one object property (e.g., cropSpecies to say for instance at what time this information was captured), the encoding of the previous management zone would be as the listing below (also available for download[58]). Note that no extra properties are attached in the example though, as this is just for illustration).

```
{
 "@context": [
    "https://rapw3k.github.io/DEMETER/DEMETER-agricontext.jsonld",
    "https://rapw3k.github.io/DEMETER/DEMETER-core-meta-model.jsonld",
    {
     "label" : "http://www.w3.org/2000/01/rdf-schema#label",
     "geometry": {
```

[57] https://github.com/rapw3k/DEMETER/blob/master/DEMETER-core-meta-model.jsonld
[58] https://github.com/rapw3k/DEMETER/blob/master/managementZone4-example-property-graph.jsonld

```
            "@id": "http://www.opengis.net/ont/geosparql#hasGeometry",
            "@type":"@id"
            }
      }
   ],
   "id": "http://w3id.org/foodie/core/ManagementZone/4",
   "type": "ManagementZone",
   "label": "ManagementZone #4",
   "code": {
      "type": "Property",
      "value": "CODA4"
    },
   "creationDateTime" : "2015-12-01T00:00:00",
   "cropSpecies" : {
       "type": "Relationship",
       "object": "http://w3id.org/foodie/core/CropType/20"
   },
   "holdingZone" : "http://w3id.org/foodie/core/Plot/1",
   "originType" : "http://w3id.org/foodie/core/OriginTypeValue/1",
   "zoneAlert" : "http://w3id.org/foodie/core/Alert/4",
   "geometry" : "http://w3id.org/foodie/core/ManagementZone/4/geometry"
}
```

Now, if we see the corresponding RDF/Turtle representation, it would be like the listing below:

```
@prefix ns0: <http://foodie-cloud.com/model/foodie#> .
@prefix ns1: <https://uri.etsi.org/ngsi-ld/> .
@prefix ns2: <http://www.opengis.net/ont/geosparql#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://w3id.org/foodie/core/ManagementZone/4>
  a <http://foodie-cloud.com/model/foodie#ManagementZone> ;
  ns0:code [
    a ns1:Property ;
    ns1:hasValue "CODA4"^^xsd:string
  ] ;
  ns0:creationDateTime "2015-12-01T00:00:00"^^xsd:dateTime ;
  ns0:cropSpecies [
    a ns1:Relationship ;
    ns1:hasObject <http://w3id.org/foodie/core/CropType/20>
  ] ;
```

```
ns0:holdingZone <http://w3id.org/foodie/core/Plot/1> ;
ns0:originType <http://w3id.org/foodie/core/OriginTypeValue/1> ;
ns0:zoneAlert <http://w3id.org/foodie/core/Alert/4> ;
ns2:hasGeometry <http://w3id.org/foodie/core/ManagementZone/4/geometry> ;
rdfs:label "ManagementZone #4"^^xsd:string .
```

That last individual, however, is not a valid OWL 2 DL definition, though, as properties are used differently from how they were defined in the referenced (re-used) ontology (FOODIE). In particular, the data type property *code* is used as an annotation/object property pointing to a blank node of type *Property*, and object property *cropSpecies* defined with range *CropType* is used as an annotation/object property pointing to a blank node of type *Relationship*. OWL 2 DL allows punning[59] allowing to use, e.g., an URI as class and as an individual, but still has limitations, i.e., a name cannot be used for both a class and a datatype and a name can only be used for one kind of property. Hence such individual would be treated as an RDF graph (OWL 2 Full). If we would like to move back to OWL DL to make use of, e.g., reasoning, we would need to convert back to lower expressivity (remove the property graphs).

Finally, it is worth noting that, similar as the NGSI-LD approach presented in annex B of [ETS6] mentioned above, DEMETER AIM will map entities from selected major ontologies/vocabularies for the cross-domain and domain layers to the core AIM meta-model.

7.2 Cross-Domain ontology

Cross-domain ontologies are defined as a set of generic models which are aimed at avoiding conflicting or redundant definitions of the same classes in the domain-specific layer. Selecting such ontologies is the basis for interoperability with other information systems and tooling that already use these, so in general “canonical” ontologies managed by standardisation bodies are preferred, although “de facto standards” in widespread use may have advantages. As part of the AIM, we plan to adopt a cross-domain ontology similar to NGSI-LD, but as opposed to the original NGSI-LD, by using standardized or widely adopted ontologies. In particular, we want to provide a cross-domain ontology that builds upon the core meta-model defined in section 7.1 and consolidates the high-level concepts occurring in domain-specific ontologies presented in section 7.3, while complementing the metadata layer presented in section 7.4. Ontology alignments between AIM components and other ontologies can increase interoperability and should be based on ontology alignments between cross-domain ontologies as a foundation.

Different cross-domain ontologies may represent high-level models found useful for different parts of the modelling problem. For example, in-situ observations, time-series, mobile devices, remote sensing, numerical modelling, statistical summaries, relationship graphs, data catalogues, etc. all have different patterns that are

[59] https://www.w3.org/TR/owl2-new-features/#F12:_Punning

most effectively and commonly handled in different meta-models, and specific cross-domain ontologies may be required for each distinct pattern in the DEMETER information scope to support efficient processing, even if a flexible common meta-model may be used for data transfer operations.

The NGSI-LD specification proposes a small set of cross-domain terms on e.g. geometric and temporal properties. Complementary to the concepts proposed in the NGSI-LD specification, ontologies for the description of sensor data should be employed on the cross-domain layer, as sensors and actuators play a role in virtually all domains served by DEMETER (see Appendix I). Secondly, an ontology for describing statistical data sources should be employed in the cross-domain layer of AIM. Ontologies for the description of general properties of "digital artefacts" such as tabular data, however, are covered in the metadata part of the model, described in section 7.4.

Implicit in this is the need for means to manage extension mechanisms. Groups such as FIWARE implement shared repositories for data models and tooling to check the conformity with some basic documentation standards. DEMETER should adopt something similar, but also recognise that the FIWARE model (of a long flat list) will not scale to handle multiple intersecting communities of practice, or the proposed DEMETER approach of factoring out common elements. In practice, the NGSI layered model represents the recognition that such refactoring needs to happen at different levels and may in fact be nested in as many levels as makes sense to separate concerns across related sub-groups.

In the following, some concrete standards or de-facto standards which should be re-used in the AIM cross-domain layer are presented.

7.2.1 Cross-Domain Ontologies and Vocabularies

7.2.1.1 NGSI-LD

NGSI-LD defines several cross-domain concepts, namely, geographical and geometrical properties, temporal properties and time values, and unit-code properties. They all are of relevance for the AIM model but, apart from a JSON type property, do not convey any taxonomic information or constraints on value ranges and are also not bound to any standards or well-known ontologies. Therefore, we propose supplementary ontologies which should serve as a replacement for NGSI-LD terms when adopted in the DEMETER AIM model, although at the same time we propose also to define mappings with the NGSI-LD terms in order to enable the interoperability between the two models.

7.2.1.2 Temporal Properties and Time Values

W3C OWL Time[60] [OWL17] conveys temporal information and time values as instants or intervals, which is similar to the NGSI-LD time properties and values. As W3C OWL Time has been adopted in the SSN/SOSA ontology (see Sensors and Actuators paragraph below), but the OGC also has an active working group and a profile of the Observations and Measurements model that underpins SSN/SOSA[61]. Through this, more complex time related models such as timeseries may be proposed for standardisation through the OGC Time Domain Working Group, if necessary.

7.2.1.3 Geo Properties and Geometry Property

OGC GeoSPARQL[62] is a geographical query language, which offers a comprehensive implementation specification and has been adopted by several tools already. It comes with a definition of supported geographical and geometrical linked data representation to which the DEMETER AIM model should adhere. The GeoSPARQL specification is currently being considered for revision by OGC, so any limitations or desired extensions discovered during DEMETER can be addressed in a locally updated copy and proposed for the next version of the standard [Geo12].

7.2.1.4 Units of Measurement

QUDT[63] is one of the most comprehensive units-of-measurement ontologies [QUDT20] and compatible with SSN/SOSA. It is intended as a supplement for the NGSI-LD unitCode property.

7.2.1.5 Sensors and Actuators

The W3C, jointly with the OGC, put forward the SOSA (Sensor, Observation, Sample, and Actuator) and SSN (Semantic Sensor Network Ontology) standard, which is currently in the state of a technical recommendation.

"The Semantic Sensor Network (SSN) and Sensor, Observation, Sample, and Actuator (SOSA) ontologies are set out to provide flexible but coherent perspectives for representing the entities, relations, and activities involved in sensing, sampling, and actuation. SOSA provides a lightweight core for SSN, whereas SOSA acts as minimal interoperability fall-back level, i.e., it defines those common classes and properties for which data can be safely exchanged across all uses of SSN, its modules, and SOSA." [SSN17]

SOSA/SSN, hence, supports different perspectives: observation, actuation, and sampling, which are generally all relevant for DEMETER (see Appendix I). However, the integration should particularly focus on the observation perspective as this is most prevalent use cases across the pilots. Figure 32 below depicts exemplarily the observation perspective of SSN/SOSA. Further insights and details about the standard are outlined in Section 5.8.

[60] https://www.w3.org/TR/owl-time/
[61] http://www.ogc.org/standards/tsml
[62] https://www.ogc.org/standards/geosparql
[63] https://qudt.org/

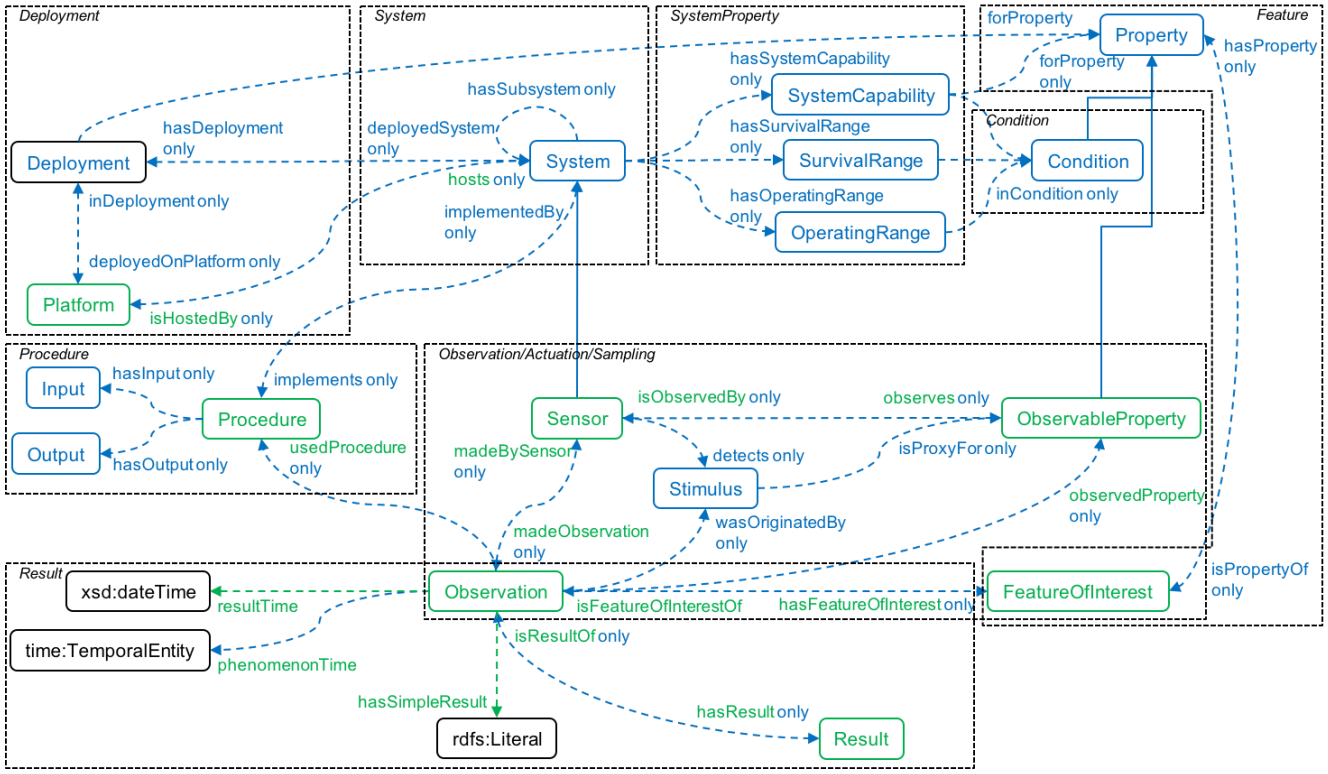


Figure 32. Overview of the SSN classes and properties (observation perspective)

SOSA provides a basic pattern for metadata for observation processes and can be used to standardise a set of properties in the "property graph" meta-model. Applications of SOSA require the development of approaches for describing *ObservableProperty* and *Procedure* sub-classes, and instances. These two classes can be used in different levels of abstraction:

E.g., consider one of the GCOS "Essential Climate Variables"[64]

ID	Area	Variable	Procedure
10106	Atmosphere	Surface Wind Speed and Direction	Wind speed over ocean surface (horizontal)

If the sensor measures wind speed, then:

ObservedProperty: "wind speed"

[64] https://www.ncdc.noaa.gov/gosic/gcos-essential-climate-variable-ecv-data-access-matrix

Procedure: “measured as horizontal component at ocean surface.”

Procedures also tend to have spatio-temporal sampling distributions inherent in sample design.

Determining how these concerns are modelled and how finely grained descriptions are generated will be a significant activity regardless of which cross-domain model and domain specific models are chosen - there is always a choice of the boundaries between model, controlled vocabulary and descriptive text elements that need to be identified.

Integration process of SSN/SOSA with DEMETER AIM for ObservedProperty and Procedure

The following steps are necessary for the integration of SSN/SOSA with the DEMETER AIM:

1) Mapping standard SSN/SOSA metadata properties to property names in the AIM meta-model, based on NGSI-LD.
2) Defining appropriate subclasses of *ObservedProperty* and *Procedure* for agri-domain specific terms and mapping the properties of these sub-classes to the property graph model defined by the NGSI-LD meta-model that is extended by AIM.
3) Determining controlled vocabularies elements for specific elements of NGSI-LD data models that cannot be covered by the steps above.
4) Mapping NGSI-LD data structures onto equivalent common models using controlled vocabularies for constant elements. For example: a schema used by some given source of NGSI-LD data may be specifically for air temperature readings, without any specific element stating that “air temperature” is the observed property, or information on the property observed may be contained only implicitly in metadata about the sampling procedure, etc.

7.2.1.6 Earth Observations Data

Earth Observations and derived data are typically provided by external services, and do not need to be explicitly modelled inside AIM. However, if required, for example to deliver derived products as coverages (data grids) then these may be modelled in a multi-layer approach:

1. The container – this will use the OGC Coverage abstract model, capturing basic metadata
2. The observation metadata – this can be implemented using the SOSA/SSN Observation model with constraints around parameter values (an EO profile of AIM)
3. The data record grid encoding – this is typically an encoding such as TIFF
4. Observed values encoding – these may be simple values, such as in TIFF, but it may in fact be based on classification definitions and more complex statistical models.

Thus, not specific EO model is proscribed, but capture of the metadata about EO datasets should be undertaken

using the DEMETER metadata model to assist integration and discovery.

Next steps and tasks planned:

Task	Who	What
1	OGC	Publish semantic definitions from Implementation Standard OGC 13-026r8[65]
2	DEMETER	Publish profile of OGC 13-026r8 for DEMETER choices of options
3	DEMETER	Publish observable phenomena and sensor types as SKOS concept scheme to support semantic descriptions via QB and formal profiles of SSN/SOSA (“SENTINEL-2”, “FAPAR” etc,)
4	DEMETER	Publish RDF-QB[66] bindings for phenomena
5	DEMETER	Synthesise and test profiles and alignment models for OGC WCS and OGC APIs with DEMETER AIM and NGSI-LD compatible feature models
6	OGC	Disseminate findings of DEMETER to support alignment and ongoing evolution of related OGC activities to standardise next generation OGC API profiles.

The methodology to synthesise and to test profiles for specific EO products is:

- Create an OWL model and JSON-LD context for the containing model
- Create an OWL model for the feature types in the coverage model
- Create and add JSON-LD contexts to WCS output
- Describe each EO dataset and/or access service in DCAT, referencing these data models
- Register controlled vocabularies of each of the product identification elements (“SENTINEL-2”, “FAPAR”, etc., to make these accessible to semantic models of DEMETER data flows and catalogue services.
- Create alignment mappings between coverage features and SOSA model used in AIM (map EO-centric features describing EO data to generalized observation model and property-graph metamodel used in AIM.

Implement data transformations using these alignment mappings as required to support interoperability of EO and derived data products with the rest of DEMETER data discovery, access and translation functions.

7.2.2 Intergation with DEMETER AIM

The NGSI-LD specification proposes that cross-domain ontologies/vocabularies can be mapped to the Core NGSI-

[65] http://www.opengis.net/doc/is/opensearch-eo/1.0
[66] https://www.w3.org/TR/vocab-data-cube/

LD using the JSON-LD *@context*, by mapping terms provided as string to concepts specified as URIs.

The DEMETER AIM cross-domain layer is being implemented as a suite of ontology modules that reuse in part or as whole the selected ontologies, along with the corresponding JSON schema and JSON-LD @context documents, which can be incrementally developed and tested individually and combined into high-level context documents for each data profile required.

The specifics of the implementation shall be in accordance with other layers of the AIM model, described in Sections 7.1, 7.3 and 7.4 from which a common strategy should evolve.

An extension of the cross-domain vocabularies with ad-hoc terms shall happen if and only if proven necessary. Therefore, required terms and axioms are added and aligned with the cross-domain ontology in analogy with the Integration process of SSN/SOSA with NGSI-LD for ObservedProperty and Procedure explained above in Section 7.2.1.

7.3 Domain-Specific ontologies

7.3.1 Main principles and rationale

In addition to the cross-domain ontology that will cover the data model that spans all the different application domains, we also need to provide ontologies (as part of AIM) that will cover the data needed to be stored in each individual domain. Following the requirement analysis that we performed, we have identified a number of domains that the DEMETER AIM needs to cover. As our goal is to ensure interoperability with existing ontologies and systems, we base this model on a number of such existing ontologies with their terms aligned (as several have overlapping terms) and enriched (where appropriate).

We want to point out that the requirements we collected regarding the data models and modelling (which were presented in section 6.1) are driving the development of the domain-specific ontologies. This is in line with the *NeOn methodology* [SuGo12]. The overall aim of the requirements specification process is to state why the ontology is being built and to identify and define the requirements the ontology should fulfil, including to define the purpose and scope of the given ontology or ontology module and to provide the ontology development team with the necessary documentation about the domain to be modelled. Then, the ontology implementation process takes place in order to build the ontology using a formal language, based on the ontological requirements identified by the domain experts when collecting the requirements. After defining the first set of requirements, though modification and addition of requirements are allowed during the development, the ontology implementation phase is carried out (the implementation is presented in Section 9).

In view of this process, we start by presenting here the different domains and types of data that need to be covered as they were identified by the requirement analysis. We also present the existing ontologies that provide (part of) the data needed to be represented in order to cover the needs identified by the technical requirements.

Following the requirement analysis, DEMETER's data model needs to enable a common representation of agronomic data including: farm data and data about farm economics; field data: e.g. the location and geometry of the fields; data regarding the irrigation and fertilization of fields; soil data such as soil temperature and moisture, soil physical and chemical analysis; data about crops and their treatment, e.g. crop type, crop developing stages, crop cultivar or variety, crop health status, pests and pesticides; data related to sensors' measurements such as carbon content. We have identified that most of these data are represented in existing ontologies and we based our data model primarily on the following ontologies: saref4agri[67], FIWARE[68] for smart agri, Foodie[69] and SSN[70]/SOSA[71]. More specifically, a number of classes and concepts are relevant to our model and, in fact, there are overlaps with between the relevant terms in these ontologies, as seen in the following analysis. The following classes from various ontologies are relevant for the data needed to be represented:

- For crop data: FOODIE cropSpecies and cropType; FIWARE AgriCrop, Agrifood; Saref4agri s4agri:Crop and s4agri:PlantGrowthStage.
- For crop pests: FIWARE AgriPest, Foodie Pest.
- For farm location and farm parcel/plot data: FIWARE AgriFarm, AgriParcel; FOODIE Farm, Plot and Management Zone.
- For watering and fertilization data: FOODIE Fertilization; Saref4agri s4agri:WateringSystem.
- For soil data: FIWARE AgriParcelRecord (with many attributes e.g. soilTemperature, soilMoistureVwc, soilSalinity etc); Saref4agri s4agri:SoilMoisture, s4agri:SoilTemperature.

In addition, these can then be aligned with terms from the SSN cross-domain ontology and enriched with extra concepts/tems from AGROVOC [72] regarding crops, or plant products or pests, using the `agroVocConcept` property (which we can use similarly to how FIWARE uses it currently in order to connect to Agrovoc entities), where necessary.

In these ontologies, FIWARE tends to have the most attributes associated with each concept (class), however the ontology is still work in progress and its models are available in one large (mostly flat) FIWARE JSON-LD context. On the other hand, Foodie has the most classes, but with relatively less documentation and attributes. Saref4Agri has a decent number of concepts and also has very good documentation so it is a promising start. Thus, the general approach was to use SAREF4Agri as main source given its good documentation, structure and coverage, and extend with FOODIE and FIWARE entities as well as entities from other ontologies where needed.

The domain specific ontologies for all these domain as described in the subsections that follow. Here we only

[67] https://mariapoveda.github.io/saref-ext/OnToology/SARE4Agri/ontology/saref4agri.ttl/documentation/index-en.html
[68] https://github.com/smart-data-models/dataModel.Agrifood
[69] http://agroportal.lirmm.fr/ontologies/FOODIE
[70] http://www.w3.org/ns/ssn/
[71] http://www.w3.org/ns/sosa/
[72] http://aims.fao.org/standards/agrovoc/linked-data

summarize what each one covers.

First, the domain that deals with Parcel/Plot data, which describes a plot of land used to plant crops and which is part of a farm. Now, all three ontologies (FIWARE, Saref4agri and Foodie) have an equivalent class for this. So the Saref4Agri concept was aligned with the equivalent ones from Foodie, while also extending with extra properties that are present in the AgriParcel class of FIWARE Agri. This was done to align the properties and remaining classes defined.

Second, the next specific domain data regards a common representation of livestock data (and this will be useful when we discuss traceability of products). Again, we reused the information regarding data for parcels and farms used above (in farm data), but now each parcel is used for raising livestock. In addition, we need to include data regarding the animals and any sensors (e.g. wearable) on them; regarding milk and meat production and quality, milk properties and quality. In addition, information such as livestock number, birth date of the animal, sex, weight, or species. We also need to model other types of animals such as poultry, apiary and hives. A number of classes and concepts are relevant to our model from existing ontologies as seen in the following analysis. For animal data the following classes were relevant from various ontologies: FIWARE Animal; Saref4agri s4agri:Animal, s4agri:MilkingSensor, s4agri:ActivitySensor; INSPIRE: Animals and animals health. Regarding sensors and wearables on animals we also take concepts from SSN as well in addition to the s4agri classes.

Third, the next domain needs to enable a common representation of agricultural machinery data such as: engine data, fuel consumption, emissions etc., as well as information about the machines such as synchronous speed, mechanics scheme, rotational speed, synchronous pull-out torque, etc. For this domain, relevant ontologies (in addition to general ontologies that deal with sensors such as SSN) are the FOODIE Transport data model and the AFarcloud hierarchy of robotic vehicles (UGVs, UAVs).

Fourth, DEMETER needs to enable the representation of current earth observation (EO) data as well as historical EO data. For this we use OGC GeoSPARQL[73] and its connection with Saref4Agri as we are already using SAREF as basis for a number of other specific domain ontologies. However, as this spans all the domains it has been decided that this requirement to be handled in the cross-domain ontology and it has already been presented in Section 7.2.1.6.

Fifth, DEMETER needs to enable the representation of weather data (e.g., temperature, humidity, wind speed/direction, solar radiation, pressure, sound pressure, sound intensity) and open spatial data modelling. For this we based it on the relevant ontologies from FIWARE together with the classes and attributes present in saref4agri: FIWARE Weather Observed, Weather Forecast, Weather Alert and Saref4agri s4agri:AmbientHumidity, s4agri:AirTemperature.

Sixth, DEMETER needs to include food traceability information of a number of products such as dairy products

[73] https://www.ogc.org/standards/geosparql/

and pastries or poultry products (production`, transport, retail) to be used in the product passport information. For this we took some concepts and terms from FOODON; however this introduces a huge ontology, and thus, for the time being, we decided to include only some of its main terms and include GPS locations (and not state/country info as in FOODON) using the WGS84 Geo Positioning ontology[74] instead. Moreover, the GS1 EPCIS standard[75] is being investigated to be exploited by the DEMETER AIM in support of agrifood product traceability.

Finally, DEMETER needs to enable a common data model able to interpret farmers' needs and preferences including: farmers' needs related to cost optimization (e.g. linking economical aspects of wholesale and retail prices), production issues (better quality of their products, crop variety per field, optimal date for planting and harvesting), cost/benefit analysis of field operations (irrigation/fertilization), optimization on irrigation/fertilization strategies, disease monitoring, yield analysis (e.g. the estimation of crop yield according to climate conditions), animal welfare tracking; production preferences (e.g. the use of non-chemical pesticides, attention to animal welfare, transparency to the consumers), any other relevant data input collected during farm operations (related to animal welfare, crop production, product's characteristics). So far we are not aware of any dominant ontology that models such data, and we have not included this domain in the current version of the DEMETER AIM; it looks like such an ontology would have to be developed for this project depending on the needs of the farmers identified in the pilots and included in the final version of the DEMETER AIM in deliverable D2.3.

7.3.2 Agriculture Profile ontology

The Agriculture Profile ontology is available under https://w3id.org/demeter/agri. This imports all the remaining ontologies used in AIM and is briefly presented below.

```
@prefix: < https://w3id.org/cybele/> .
@prefix qb: < http://purl.org/linked-data/cube#> .
@prefix dct: < http://purl.org/dc/terms/> .
@prefix owl: < http://www.w3.org/2002/07/owl#> .
@prefix rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: < http://www.w3.org/XML/1998/namespace> .
@prefix xsd: < http://www.w3.org/2001/XMLSchema#> .
@prefix dcat: < http://www.w3.org/ns/dcat#> .[76]
@prefix foaf: < http://xmlns.com/foaf/0.1/> .[77]
@prefix prov: < http://www.w3.org/ns/prov#> .
@prefix rdfs: < http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: < http://www.w3.org/2004/02/skos/core#> .
@prefix stat: < http://data.europa.eu/m8g/> .
```

[74] https://www.w3.org/2003/01/geo/wgs84_pos
[75] https://www.gs1.org/standards/epcis
[76] https://www.w3.org/TR/vocab-dcat-2/
[77] https://en.wikipedia.org/wiki/FOAF_(ontology)

```
@prefix schema: < http://schema.org/> .
@prefix af - inspire: < http://inspire.ec.europa.eu/schemas/af/3.0#> .
@prefix act - inspire: < http://inspire.ec.europa.eu/schemas/act-core/3.0#> .
@prefix foodie: < http://foodie-cloud.com/model/foodie#> .
@prefix saref4agri: < https://w3id.org/def/saref4agri#> .
@prefix common: < http://portele.de/ont/inspire/baseInspire#> .
@prefix fiware: < https://uri.fiware.org/ns/data-models#> .
@prefix iso19109: < http://def.seegrid.csiro.au/isotc211/iso19109/2005/feature#> .
@prefix iso19150 - 2: < http://def.seegrid.csiro.au/isotc211/iso19150/-2/2012/basic#> .
@prefix iso19103: < http://def.seegrid.csiro.au/isotc211/iso19103/2005/basic#> .
@prefix geo: < http://www.opengis.net/ont/geosparql#> .
@prefix saref: < https://w3id.org/saref#> .
@prefix ssn: < http://www.w3.org/ns/ssn/> .
@prefix obo: < http://purl.obolibrary.org/obo/> .
@base < https://w3id.org/demeter/> .

< https://w3id.org/demeter/agri> rdf:type owl:Ontology ;
       owl: versionIRI <
https://raw.githubusercontent.com/rapw3k/DEMETER/master/models/demeterAgriProfile.ttl> ;
       dct: contributor[schema:affiliation[foaf:name "OGC"];
                       foaf:name "Rob Atkinson"],
                       [schema:affiliation[foaf:name "ICCS"];
                          foaf:name "Ioanna Roussaki"];
       dct:creator[schema:affiliation[foaf:name "PSNC"];
                        rdfs:seeAlso < http://orcid.org/0000-0003-4289-4922> ;
                        foaf: name "Raul Palma"];
       owl:imports < https://w3id.org/demeter/agri/agriCommon> ,
                        < https://w3id.org/demeter/agri/agriFeature> ,
                         < https://w3id.org/demeter/agri/agriCrop> ,
                         < https://w3id.org/demeter/agri/agriIntervention> ,
                         < https://w3id.org/demeter/agri/agriAlert> ,
                         < https://w3id.org/demeter/agri/agriProduct> ,
                         < https://w3id.org/demeter/agri/agriProperty> ,
                         < https://w3id.org/demeter/agri/agriSystem> ,
                         < https://w3id.org/demeter/agri/agriPest> ,
                         < https://w3id.org/demeter/agri/farmAnimal> ,
                         < https://w3id.org/demeter/agri/agriResource> ;
        dct: description "The DEMETER Agri Profile is a master profile importing focused
specific profiles/modules of DEMETER AIM."@en;
        dct: rights "This vocabulary is distributed under Creative Commons Attribution 4.0
License - http://creativecommons.org/licenses/by/4.0"@en;
        dct: title "DEMETER AgriCrop"@en;
        rdfs: comment "The DEMETER Agriculture Information Model (AIM) is the common
vocabulary in DEMETER project providing the basis for semantic interoperability across smart
farming solutions"@en;
        owl: versionInfo "1.0";
       foaf: maker[foaf:homepage < https://h2020-demeter.eu/> ;
                    foaf: name "DEMETER project"] .
```

As can be seen in the previous depicted .ttl file, it starts with prefixes *@prefix* allowing to declare instead of a long prefix of the repeated URI with a short prefix. In the .ttl[78] file, you can also see the *@base <https://w3id.org/demeter/>* . The *@base* allows extra abbreviation of URIs, however it is often used for simplifying the URIs in the data, where the prefix directives are for vocabularies which describe the data. You can also see that this also uses the *foaf* ontology (*friend of a friend)* which is a machine-readable ontology describing persons, their activities and how they are related to other people and objects. It also defines prefixes for a number of other ontologies that are being used, including DCAT (used by the metadata schema presented in section 7.4), FOODIE, saref4agri, FIWARE and SSN.

The demeterAgriProfile ontology imports (and therefore consists) of the following ontologies: *agriCommon, agriFeature, agriCrop, agriIntervention, agriAlert, agriProduct, agriProperty, agriSystem, agriPest, farmAnimal.* The initial/core classes are the following: *ActivityComplex, Agent, Agri Farm, Agri Parcel, AgriParcelOperation, AgriParcelRecord, AgriPest, AgriProductType, Alert, Animal, AnyFeature, Codelist, Datatype, Deployment, EconomicActivityNACEValue, Feature of interest, FeatureType, ID, Measure, Measurement, Period, Platform, Property, skos:Concept, SpatialObject, System, taxonomic_rank, Unit of measure.*

7.3.3 Agriculture Commons ontology

The Agriculture Commons ontology is available under https://w3id.org/demeter/agri/agriCommon and is presented in the figure below.

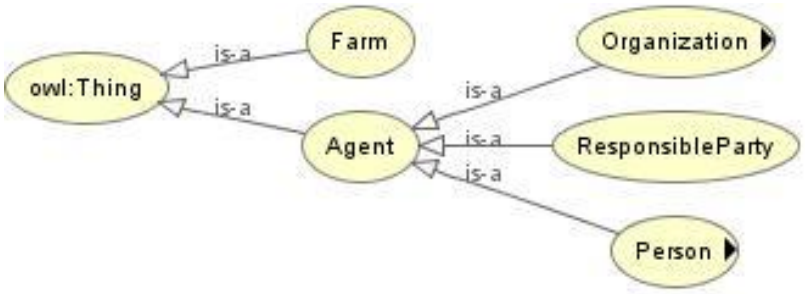


Figure 33. Visualization of the Agriculture Commons ontology

In addition, a number of basic classes and concepts are defined in this module: *Agent* and its subclass *Person* are taken from the FOAF ontology and a subclass *Farmer* is defined as well for use in the Demeter ontology. In the same manner, the general class *Organization* is subclassed by the *FarmHolding* class.

[78] https://www.w3.org/TeamSubmission/turtle/

7.3.4 Agriculture Features ontology

The Agriculture Features ontology is available under https://w3id.org/demeter/agri/agriFeature and is briefly presented below.

The agriFeature.ttl consists of the following classes: *ActivityComplex* class, with the following subclasses: *Farm* class (defined by Saref4agri) which defines a plot of land that is used for the scope of farming, containing buildings and parcels, and *Holding* class. *Agri Farm* class (defined by Fiware) which describes a generic farm constituting of buildings and parcels. *Agri Parce*l which describes the conditions recorded in a generic greenhouse, with subclass *Agri Greenhouse*. *AnyFeature* class, with subclasses: *Farm* class (defined by Saref4agri) which is used for farming containing buildings and parcels, *Holding* class, *ManagementZone* class (defined by Foodie), *Parcel* class (defined by Saref4agri) which is an area of land that cannot be divided and contains homogeneous items, *Plot* class (defined by Foodie) and *Site* class. *Codelist* (defined by Foodie) class, with *OriginTypeValue* (defined by Foodie). *Crop* class (defined by Saref4agri), *EconomicActivityNACEValue* class, *FeatureType* class, with subclasses: *Farm*, *Holding*, *ManagementZone*, *Parcel*, *Plot* and *Site*. *MachineType* class, *PropertyType* class, *skos:Concept* class with subclass: *OriginTypeValue* (defined by Foodie). *SpatialObject* class which represents everything that can have a spatial representation, with subclasses: *Feature* which represents the top-level feature type and it is similar to GFI_Feature of ISO 19156:2011, and subclasses: *Agri Farm* class, *Agri Parcel* class, (with subclass *Agri Greenhouse*), *Building* class (defined by Saref4agri) which represents a structure providing shelter for its occupants or contents and stands in one place, *Building space* class (defined by Saref4agri) which is used to define the physical spaces of the building, *Farm*, *Holding*, *ManagementZone*, *Parcel*, *Plot* and *Site*. *Geometry* class which is equivalent to the UML class GM_Object defined in ISO 19107 with subclass: *Point* which describes a point using a specific coordinate system for instance WGS84. The last class is *TractorType*.

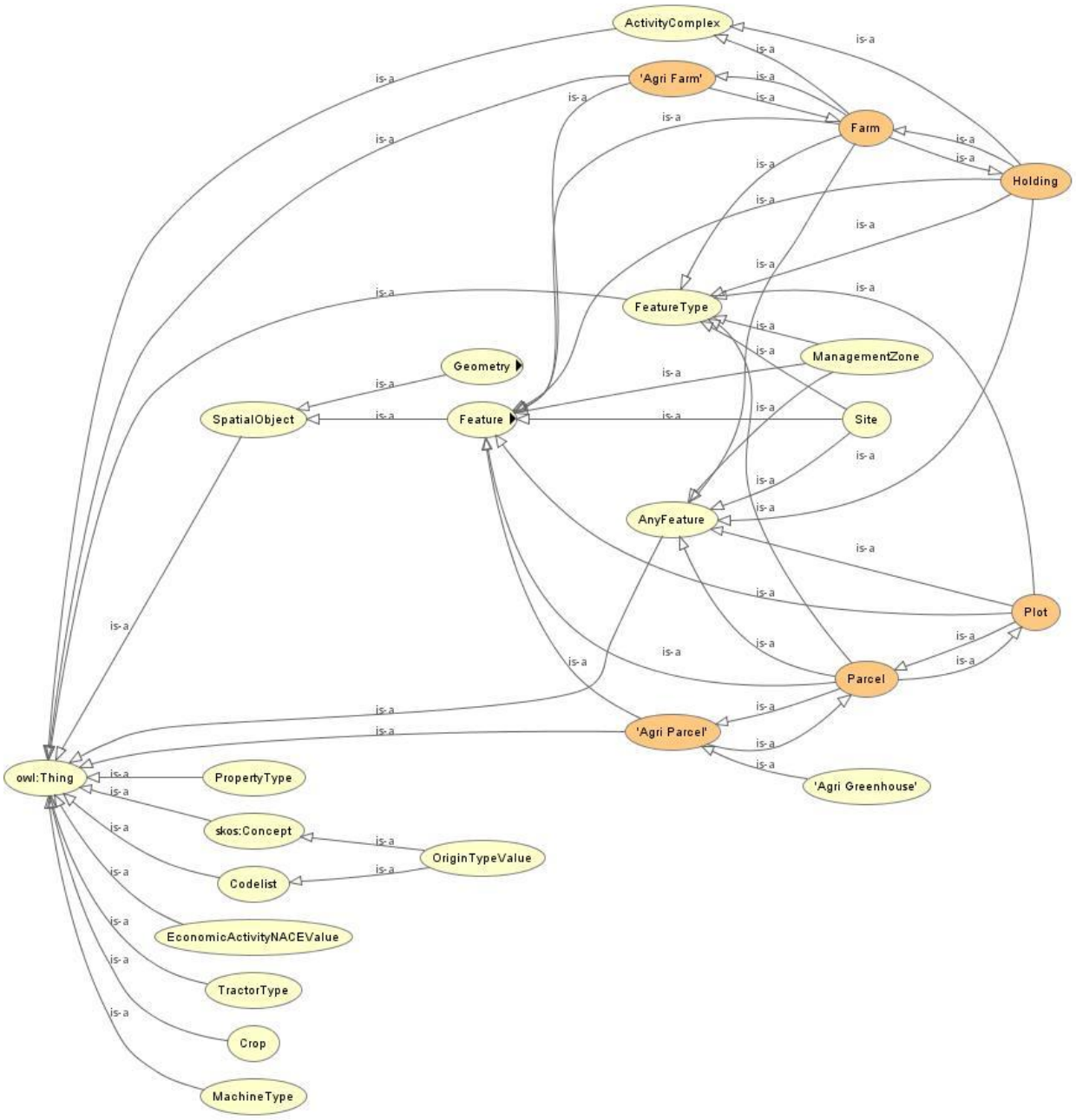


Figure 34. Visualization of the Agriculture Features ontology

Concerning the Object properties of this ontology, the following are used: *Activity*, *Contains* with subclasses: "Contains" with subclasses: *contains*, *containsPlot*, *containsZone*, *hasAgriParcel*, *hasAgriParcelChildren*. *Crop property*, *geo:location*, *has geometry* with subclasses: *landLocation*"and *Location*. *hasAgriCrop* property, *hasAgriSoil*, *hasDevice*, *includesAnimal*. Property *is contained in* with subclasses: *hasAgriParcelParent*,

holdingPlot, *holdingSite*, *holdingZone*, *Machine* property, *originType*, *soilProperty*, *Tractor* and *Within*.

The data properties that are used, are being described below: *Area* which shows the area of the parcel nominally in square meters, *Category* which shows the category of the parcel of land, if it is arable, grassland, vineyard, orchand, mixed crop, etc., *Code* (defined on Foodie), *createdAt*, *CropStatus* which describes the crop planting status such as seeded, justBorn, growing, maturing, readyForHarvesting, *Description* (defined by Foodie), *has serialization* which makes the connection between a geometry object with its text-based serialization, *hasName* (defined by Saref4agri), *lastPlantedAt* which indicates when the crop was last planted, *Notes* (defined by Foodie), *prov:generatedAtTime*, *prov:invalidatedAtTime*, *validFrom*, *validTo*.

7.3.5 Agriculture Crops ontology

The Agriculture Crops ontology is available under https://w3id.org/demeter/agri/agriCrop and is briefly presented below.

This entity contains a harmonised description of a generic crop. This entity is primarily associated with the agricultural vertical and related IoT applications.

There are three equivalent classes named *AnyFeature*, *Feature* and *FeatureType* that encapsulate the necessary subclasses enabling interoperability among existing ontologies. The aforementioned subclasses are *Crop* and *CropSpecies. Datatype* class has the *CropType* and *ProductionType* subclasses that define the types of crop holding specific cardinality restrictions in most properties. Note that instances of *CropType* can be linked to the concepts in AGROVOC using the *agroVocConcept* property from FIWARE. Other classifications may be also linked in future releases. *Geometry, measure, measurement, Property, PropertyType* are other classes defined in this model, used mostly for ranging purposes.

Object Properties are relationships defined between class objects. A number of such properties (*cropArea, cropHasAgriSoil,* cropSpecies, *hasAgriPest, hasAgriFertiliser, hasRank*) refer to crop features, while others to production (*production, productionAmount, productionProperty*). The rest deal with properties of crop (*has feature of interest, has property* and their inverse *is feature of interest of* and *is property of,* respectively).

Likewise, Data Properties match an object to a value and not another object. These properties are string-typed *family*, *code*, *genus*, *createdAt*. d*escription, harvestingInterval, name, notes, species, variety and* the datetime properties *plantingFrom, has plant date, validFrom* and *validTo,* the equivalent *productionDate* and *has harvest date,* as well the provenance ones (*generatedAtTime, invalidatedAtTime*). There is also an enumeration, named *wateringFrequency*.

Figure 35. Visualization of the Agriculture Crops ontology

7.3.6 Agriculture Interventions ontology

The Agriculture Interventions ontology is available under https://w3id.org/demeter/agri/agriIntervention and is briefly presented below.

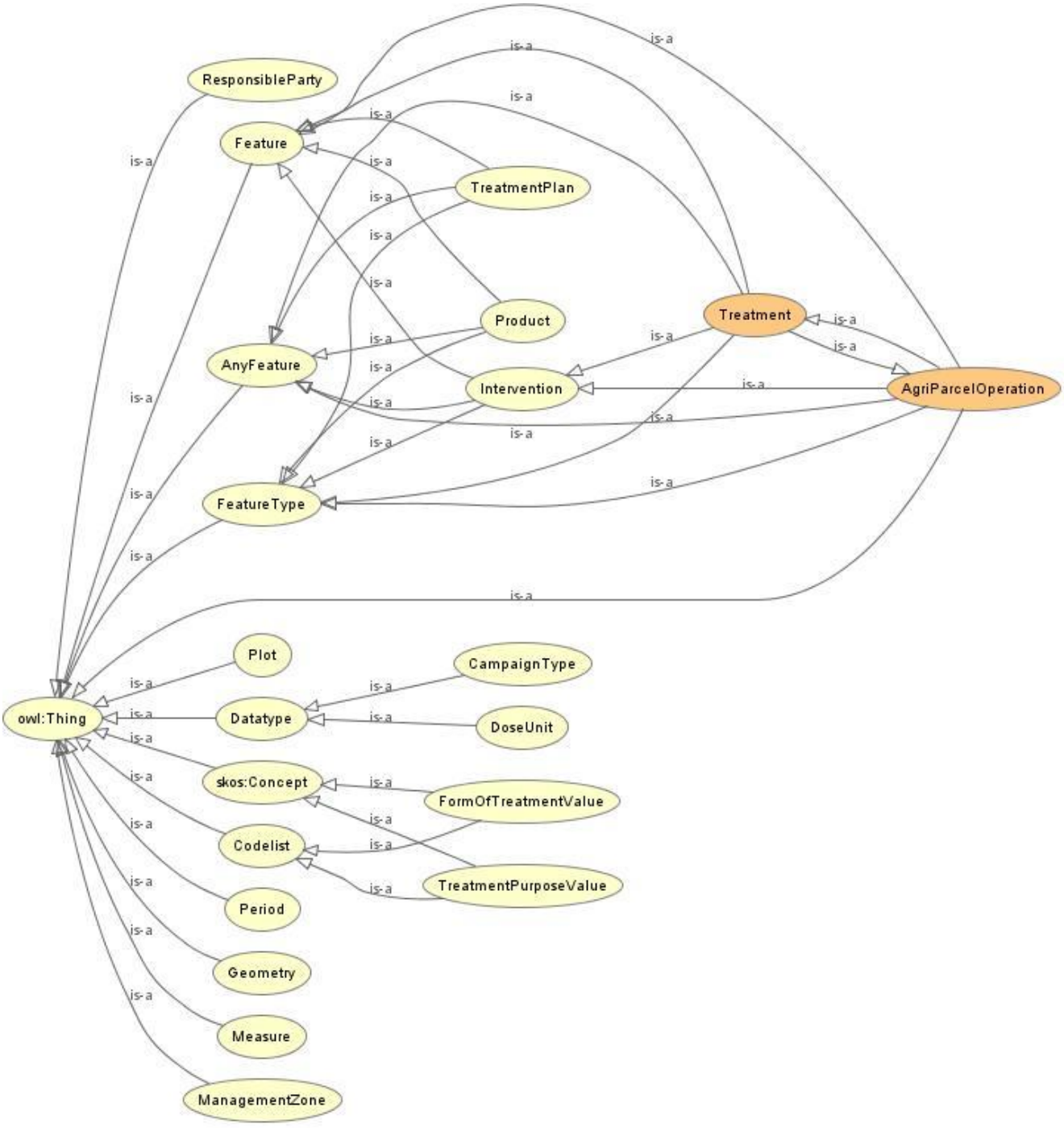


Figure 36. Visualization of the Agriculture Interventions ontology

This entity contains a harmonised description of generic operations performed on a parcel of land. This entity is

primarily associated with the agricultural vertical and related IoT applications.

There are three equivalent classes named *AnyFeature*, *Feature* and *FeatureType* that encapsulate the necessary subclasses enabling interoperability among existing ontologies. The aforementioned subclasses are the equivalent *AgriParcelOperation* and *Treatment*, *TreatmentPlan* and *Product. Datatype* class has the *CampaignType* and *DoseUnit* subclasses that define a number of cardinality restrictions on properties, while the equivalent *Concept* and *Codelist* encapsulate the *FormOfTreatmentValue* and *TreatmentPurposeValue* subclasses. Note that *Product* may be connected also to AGROVOC concepts like pesticides types, fertilizers types, and in the future we may also connect other classifications. M*easure, Period, ResponsibleParty, ManagementZone, Geometry and Plot* are more classes defined in this model, used for ranging purposes.

Object Properties are relationships defined between class objects. This model basically matches the operation to the corresponding data, as defined by the classes discussed previously. The relations defined are *applicationWidth, campaign, evidentParty, flowAdjustment, formOfTreatment, interventionGeometry, interventionZone, maximumDose, minimumDose, motionSpeed, period, plan, planProduct, pressure, quantity* and *supervisor*. The rest deal with properties of the intervention (*hasOperator, hasAgriProductType, operationHasAgriParcel* and their equivalent *operator, treatmentProduct* and *interventionPlot,* respectively).

Likewise, Data Properties match an object to a value and not another object. These properties are the descriptive (literal or numerical) *price*, d*escription, quantity, treatmentDescription, treatmentPlanCode, treatmentPlanCreation and* the datetime properties *creationDateTime*. *reportedAt, validFrom* and *validTo, plannedStartAt* and *plannedEndAt.* There are also some enumerations (*result, status, operationType, waterSource*).

7.3.7 Agriculture Alerts ontology

The Agriculture Alerts ontology is available under https://w3id.org/demeter/agri/agriAlert and is briefly presented below.

The purpose of this model is to support the generation of notifications for a user or trigger other actions, based on alerts. An alert is generated by a specific situation. The main features of an alert are that it is not predictable and that it is not recurrent data. That means that an alert could be, for example, an accident or an extremely high level of measure.

There are three equivalent classes named *AnyFeature*, *Feature* and *FeatureType* that encapsulate the necessary subclasses enabling interoperability among existing ontologies. The aforementioned subclasses are *Alert*, *CropSpecies*, *ManagementZone* and *Plot*. *Geometry* is another class defined in this model, used for ranging purposes.

Object Properties are relationships defined between class objects. This model basically matches the alert to the

corresponding data, as defined by the classes discussed previously. Consequently, the properties are *alertPlot, alertSpecies* and *alertZone* as well as the inverses *plotAlert, speciesAlert, zoneAlert* respectively. There is also a *location* property encapsulating the *alertGeometry* relationship that draws objects from Geometry class.

Likewise, Data Properties match an object to a value and not to another object. These properties are *address*, a*lertSource*, *code*, *data*, *dateIssued*, d*escription, severity, subCategory,* the equivalent *category* and *type* as well as *validFrom* and *validTo.*

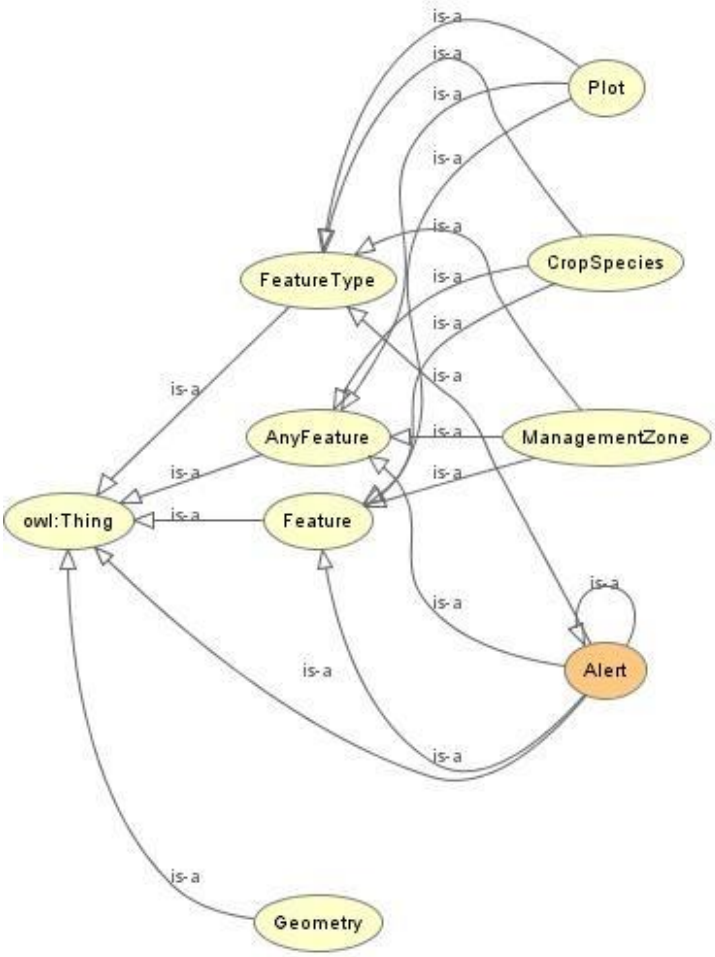


Figure 37. Visualization of the Agriculture Alerts ontology

7.3.8 Agriculture Product ontology

The Agriculture Product ontology is available under https://w3id.org/demeter/agri/agriProduct and is briefly presented below.

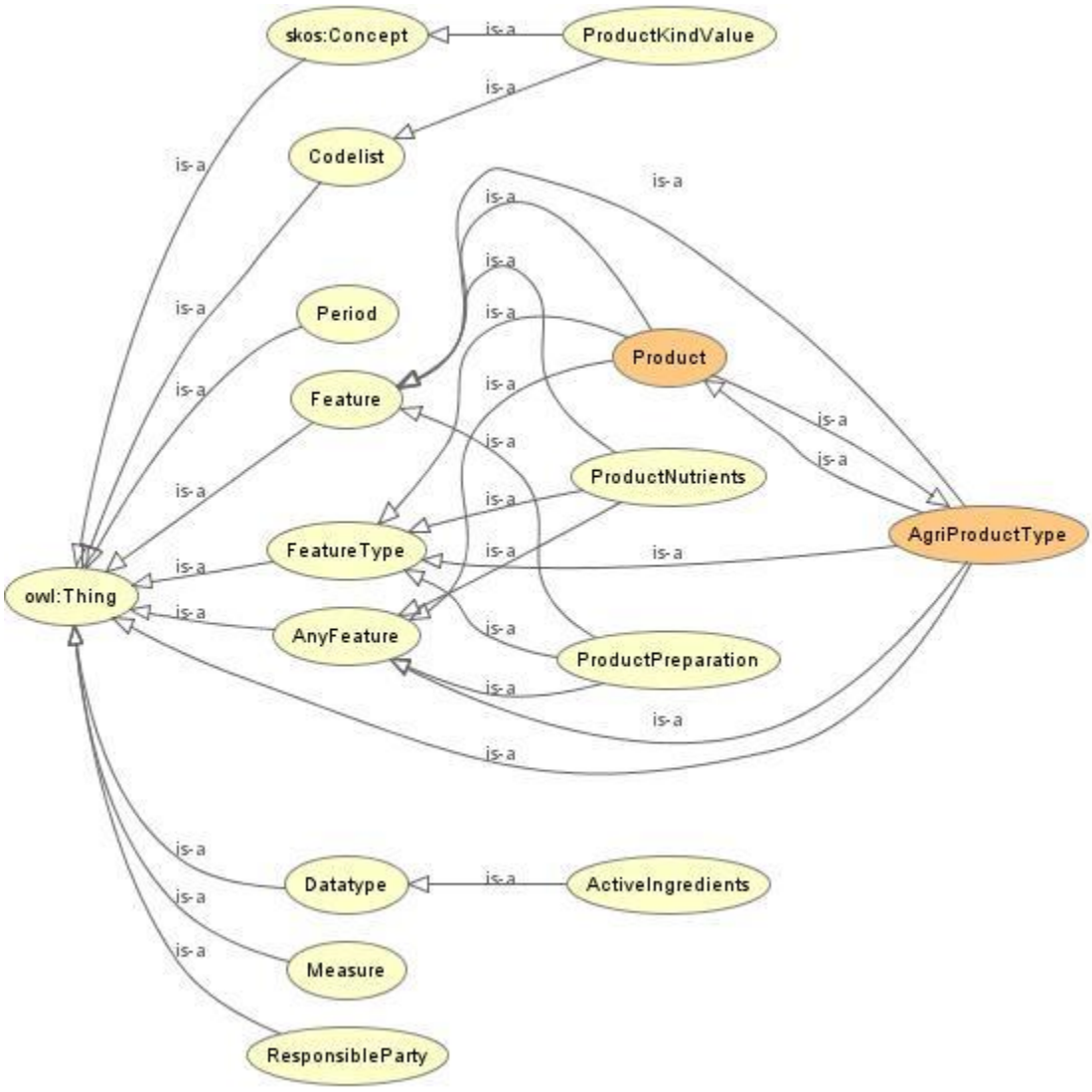


Figure 38. Visualization of the Agriculture Product ontology

This entity contains a harmonised description of a generic agricultural product type. This entity is primarily associated with the agricultural vertical and related IoT applications. The AgriProductType includes a hierarchical structure that allows product types to be grouped in a flexible way.

There are three equivalent classes named *AnyFeature*, *Feature* and *FeatureType* that encapsulate the necessary subclasses enabling interoperability among existing ontologies. The aforementioned subclasses are the equivalent *AgriProductType* and *Product*, *productNutrients* and *productPreparation. Datatype* class has the *activeIngredients* subclass and the equivalent *Concept* and *Codelist* encapsulate the *ProductKindValue* subclass. M*easure, Period, ResponsibleParty* are more classes defined in this model, used for ranging purposes.

Object Properties are relationships defined between class objects. This model basically matches the product to the corresponding data, as defined by the classes discussed previously. The relations defined are *ingredientAmount, manufacturer, nutrient, nutrientAmount, nutrientProduct, productKind, productQuantity, safetyPeriod, solventQuantity*. The rest deal with properties of the product (*hasAgriProductTypeChildren, hasAgriProductTypeParent*).

Likewise, Data Properties match an object to a value and not to another object. These properties are string-typed: *code*, d*escription, name (ingredientName, nutrientName, productName), productCode, productSubType, productType, registrationCode, safetyInstructions* and *storageHandling*, or numerical properties, such as: *nutrientMeasure* and *price,* logical *root* and the format-specific *registerUrl.*

7.3.9 Agriculture Properties ontology

The Agriculture Properties ontology is available under https://w3id.org/demeter/agri/agriProperty and is briefly presented below.

The agriProperty.ttl is consists of the following classes: *AgriParcelRecord* class which contains a harmonised description of the conditions recorded on a generic parcel of land. This entity is primarily associated with the agricultural vertical and related IoT applications. *Codelist*, which contains the subclass *PropertyTypeValue* , part of the foodie ontology. *Compressibility* class that contains the instance *vapour compressibility*, the latter contains information about the level of compression that a vapour has. *Concentration* class, which contains the instance carbon content. This indicates the carbon concentration on a farm field.

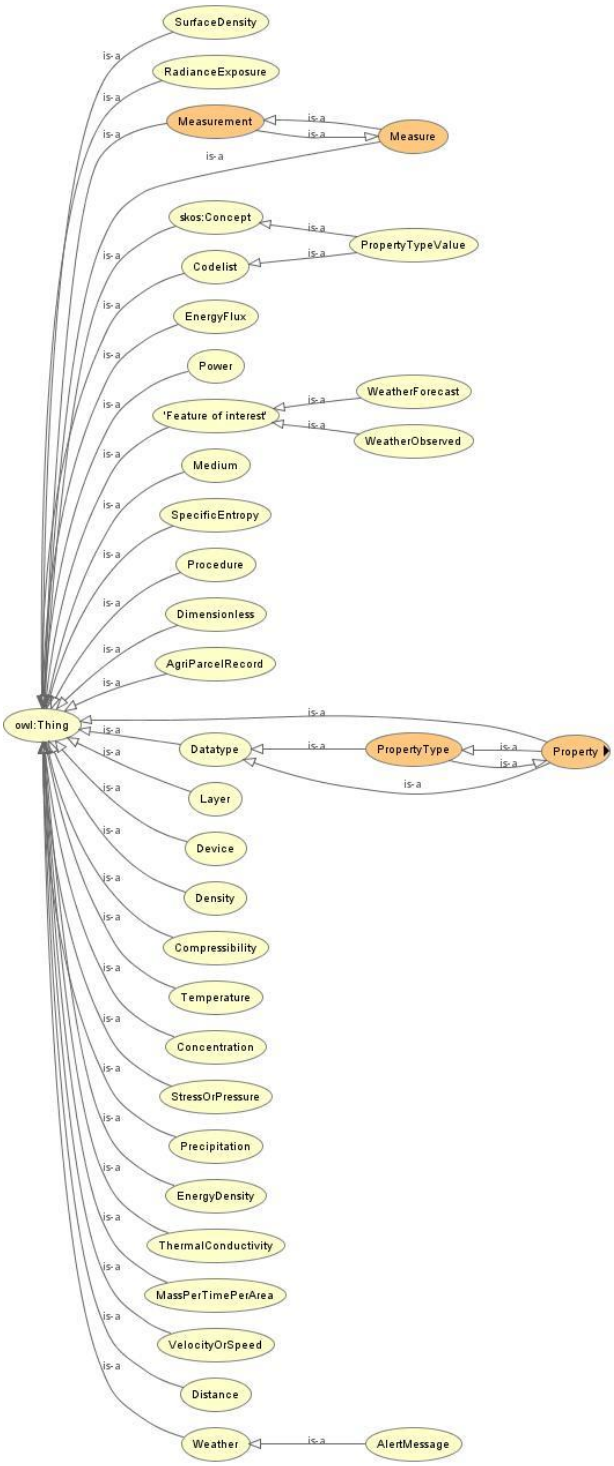


Figure 39. Visualization of the Agriculture Properties ontology

Datatype class. It contains the subclasses *Property* and *PropertyType*. *Property* class contains anything that can be sensed, measured or controlled in households, common public buildings or offices. We propose here a list of properties that are relevant for the purpose of Saref, but this list can be extended. These instances are the *Plant growth stage*, *Soil moisture*, *electricConductivity*, *pH*, *Precipitation*, *soilTextuure*, *soilType*. *Property* class contains the *Humidity* class, responsible for the instance *Ambient humidity*, and the *Temperature* class, which is responsible for the instances: *Air temperature* and *Soil temperature*. *Density* class, with the instances *snow density* and *air density*, for measuring the snow density and air density in a field. *Device* class. *Dimensionless* class, with the instances *soil albedo*, *soil porosity* and *vegetation* area fraction. *Distance* class, with the instance *snow grain size*. *EnergyDensity* class, with the instance *sound energy density, EnergyFlux*, with the instances *sound intensity in air* and *sound intensity*. *Feature of interest* class, with subclasses *WeatherForecast* and *WeatherObserved*. *Layer* class, with the instances *soil layer* and *vegetation*. *MassPerTimePerArea* with instance *snowfall flux*. *Measure* class. *Measurement* class. *Medium* class with instances: *soil pores* and *soil*. *Power* class, with instance *sound power*. *Precipitation* class with instance *snowfall*. *Procedure* class. *Property* class with subclasses *Humidity*, Temperature and instances *Plant growth stage*, *Soil moisture*, *electricConductivity*, *pH*, *Preciptation, soilTexture*. *RadianceExposure* class. *skos:Concept* class with the subclass *PropertyTypeValue*. *SpecificEntropy* class with instance *soil thermal capacity*. *StressOfPressure* class with the following instances; *sound pressure in air*, *sound pressure*, *vapour pressure*, *soil sunction at saturation*. *SurfaceDesnity class*, which consists of the instances *snow soot content*, *snowfall amount*, *atmosphere mass content of carbon dioxide, atmosphere*, *content of carbon monoxide*, *atmosphere water vapor content*, *soil frozen water content*, *soil moisture content at field capacity* and *vegetation carbon content*. *Temperature* class with *snow temperature* instance. *ThermalConductivity* class with the instance *soil thermal conductivity*. *VelocityOrSpeed* class with the instance *soil hydraulic conductivity at saturation. Weather* class with subclass *AlertMessage*.

The individuals that belong to the previous classes are the following:

For soil measurement, the agriProperty ontology uses the following individuals: *Soil*, *soil layer* , *Soil moisture*, *soil pores*, *Soil temperature*, *soil_albedo*, *soil carbon content*, *soil frozen water content*, "*soil hydraulic conductivity at saturation* , *soil moisture content at field capacity*, *soil porosity*, *soil suction at saturation*, *soil temperature*, *soil thermal capacity* , *soil thermal conductivity*, *soilTexture*, *soilType*, *volume fraction of clay in soil*, *volume fraction of condensed water in soil*, *volume fraction of condensed water in soil at critical point*, *volume fraction of condensed water in soil at field capacity*, *volume fraction of condensed water in soil at wilting point*, *volume fraction of condensed water in soil pores*, *volume fraction of frozen water in soil*, *volume fraction of sand in soil*, *volume fraction of silt in soil*, *moisture content of soil layer*, *moisture content of soil layer at field capacity*, *downward heat flux in soil*, *lwe thickness of soil moisture content*, *pH*, *electricConductivity*.

As far as the atmosphere and air are concerned, the following individuals are used: *air temperature*, that measures the degree of intensity of heat present in the air, *air density*, like air pressure, decreases with increasing altitude. It also changes with variation in atmospheric pressure, temperature and humidity, *ambient humidity*, which shows the amount of water vapour in the air, *atmosphere mass content of carbon dioxide, atmosphere*

mass content of carbon monoxide, a*tmosphere water vapor content*, *temperature*, *vapour compressibility*, *vapour pressure*, *energyFlowRate*, *HeatScheme*, *degree Celsius*, *Millibar*, *Velocity*.

As far as the snow/water are concerned, we use the individuals: *lwe convective snowfall rate*, *lwe large scale snowfall rate*, *lwe snowfall rate*, *lwe thickness of convective snowfall amount*, *lwe thickness of frozen water content of soil layer*, *lwe thickness of large scale snowfall amount*, *lwe thickness of moisture content of soil layer*, *lwe thickness of snowfall amount*, *snow density*, *snow grain size*, *snow soot content*, *snow temperature*, *snowfall*, *snowfall amount*, *snowfall flux*, *frozen water content of soil layer*, *water*, *water evaporation flux from soil*, *convective snowfall amount*, *convective snowfall flux*, *mass concentration of condensed water in soil*, *thickness of convective snowfall amount*, *thickness of large scale snowfall amount*, *thickness of snowfall amount*, *large_scale_snowfall_amount*, *large scale snowfall flux* , *liquid water content of soil layer*, *precipitation*, *massPerTimePerArea*, *Millimetre*, *Pressure*, *Scalar*.

As far as the environmental noise is concerned, we use the individuals: *sound energy density*, *sound intensity*, *sound intensity in air*, *sound power*, *sound pressure*, *sound pressure in air*, *Compressibility*, *Pressure*, *decibel-milliwatts*, *Millivolt*, *MechanicsScheme* for sound pressure. *Power* of noise.

Relative to the vegetation, the following individuals are used: *normalized difference vegetation index* (NDVI) which is a graphical indicator in order to analyze remote sensing measurements usually from space platform and can identify whether the target contains live green vegetation*, vectorProperty*, *Vegetation*, *vegetation area fraction*, *vegetation carbon content*, *carbon content*, *radianceExposure*. *radianceExposure*, *Plant growth stage*, *surfaceDensity*, *Property*, *Length*, *Distance*, *Density*, *Concentration* for carbon content on the field, *Fraction* of carbon content, *PhysicalChemistryAndMolecularPhysicsScheme* of carbon content.

Concerning the Object properties of this ontology, the following are used: *controls property*, *generalQuantityKind*, *has feature of interest*. The property *has property* with subclasses *productionProperty* and *soilProperty*. The property *hasDevice* and its equivalent *devices, smartMeter*, *hasProperty*, *is controlled by device*, *is measured by device*, *is measured in*, *is property of*, *isFeatureOfInterestOf*, *IsPropertyOf*, *makes measurement*, *measurement made by*, *measures property*, *propertyType*, *propertyType*, *recordHasAgriParcel*, *refDevice*, *refPointOfInterest*. The property *relates to measurement* with subclasses: *productionAmount*, *quantitativeProperty*. And, finally, the *relates to property.*

The data properties that are used, are being described below: *analysisDate* (defined by Foodie), *dateObserved* which contains the date and time of this observation in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval, *has timestamp* (defined by Saref), *has value* (defined by Saref) , with the following subclasses: *airTemperature*, which is the observed air temperature (in the shade) nominally in degrees centigrade, *atmosphericPressure* (defined by Fiware) which shows the atmospheric pressure observed and measured in Hecto Pascals, *dewPoint* (defined by Fiware) where the dew point encoded as a number, *Illuminance* (defined by Fiware) which stores the illuminance observed measured in lux (lx) or lumens per square metre (cd·sr·m−2), *nonQuantitativeProperty* (defined by Foodie), *pressureTendency* (defined by Fiware) which

expresses the rising or falling pressure in quantitative terms or qualitative terms in values rising, falling or steady, *relativeHumidity* (defined by Fiware) which contains air's relative humidity observed, *snowHeight* (defined by Fiware) which displays the snow height observed by generic snow depth measurement sensors, expressed in centimeters. *soilMoistureEc* which is measured as Electrical Conductivity, in units of Siemens per meter (S/m), *soilMoistureVwc* which is measured as Volumetric Water Content, VWC as a percentage. 0 ≤ soilMoistureVwc ≤ 1, *soilTemperature* which is the observed soil temperature nominally in degrees centigrade, *Temperature* (defined by Fiware) which stores the observed air's temperature, *Visibility* (defined by Fiware) which contains the visibility reported as veryPoor, poor, moderate, good, veryGood, excellent, *windDirection* (defined by Fiware) which contains the wind direction expressed in decimal degrees compared to geographic North (measured clockwise), encoded as a Number. Range 0 to 360. *windSpeed* (defined by Fiware). Next is the property *Name* with subclass *propertyName* (defined by Foodie). There is the *observedAt* which indicates the time and the date that the record was observed, *weatherType* (defined by Fiware) which is the observed weather type. It is represented by a comma separated list of weather statuses, for instance overcast, lightRain. Finally, *salinity* describes the salt content of a body of water usually measured in g/mL and *totalConsumption* expresses the energy consumed by any device in kilowatt-hours.

7.3.10 Agriculture Systems ontology

The Agriculture Systems ontology is available under https://w3id.org/demeter/agri/agriSystem and is briefly presented below.

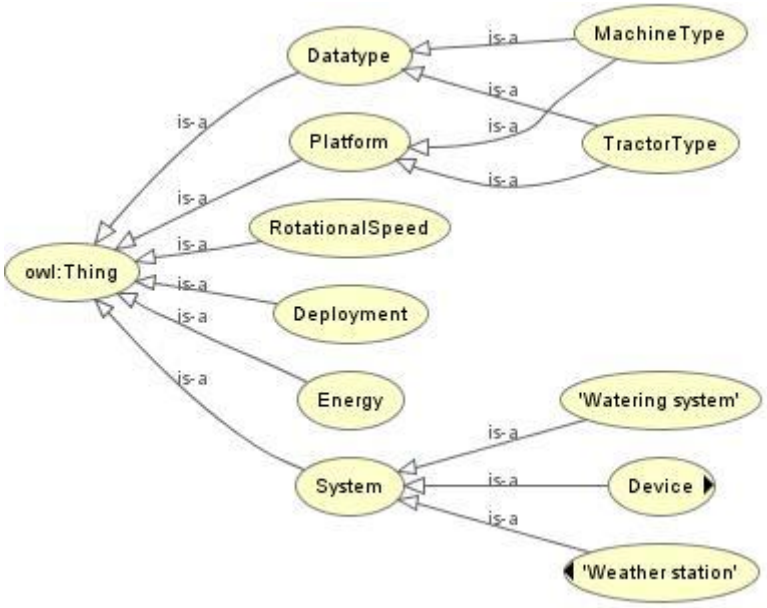


Figure 40. Visualization of the Agriculture Systems ontology

The AgriSystem.ttl ontology consists of the following classes: the class Datatype, which contains the class *MachineType* and the *TractorType* class, both of them defined by the Foodie ontology. *Deployment* class is defined by SSN ontology. *Energy* class. *Platform* class (defined by SOSA ontology) that contains the subclasses *MachineType* (defined by Foodie ontology) and *TractorType* (defined by Foodie ontology). Platform entity hosts other entities, such as Sensors, Actuators, Samplers and other Platforms. *RotationalSpeed* class, with the instances *critical build-up speed*, *critical torsional speed*, *critical whirling speed*, and *synchronous speed*. *System* class (defined by SSN ontology), which contains pieces of infrastructure that implement Procedures. Subclass of *System* class is the *Device* class (defined by SAREF ontology). The latter ontology contains the *Actuator* class (defined by SAREF ontology) and the *Sensor* class (defined by SAREF ontology). *Actuator* class contains the following subclasses: *Watering gun* class (defined by Saref4agri), which is an actuator to irrigate a space and *Watering valve* class (defined by Saref4agri). *Sensor* class contains the classes: *Eating activity sensor* (defined by Saref4agri), *Milking sensor* (defined by Saref4agri), *Movement activity sensor* class (defined by Saref4agri), *Pluviometer* class (defined by Saref4agri), which is a sensor for measuring the rain fall, *Soil tensiometer* class (defined by Saref4agri) which is a sensor for measuring the soil moisture, *Thermometer* class (defined by Saref4agri), *Weather station* class (defined by Saref4agri) which is a sensor or a system for measuring weather conditions, *Weight sensor* class (defined by Saref4agri). And finally, *Watering station* class (defined by Saref4agri).

Regarding individuals, the following are contained in the ontology: *AcousticsScheme*, *critical build-up speed* which is the lowest speed at which the machine voltage builds up under specified conditions; *critical torsional speed* which is a rotational speed at which the amplitudes of the vibrations of a machine rotor due to shaft torsional vibration reach their maximum values; *critical whirling speed* which is a rotational speed at which the amplitudes of the vibrations of a machine rotor due to shaft whirling vibration reach their maximum values, *MechanicsScheme*, *PeriodicAndRelatedPhenomenaScheme*, *property*, *rotationalSpeed*, *synchronous pull-out torque* which is the maximum torque that a synchronous machine can develop without loss of synchronism while operating at rated voltage, frequency and excitation, *synchronous speed* which is a rotational speed that results from the frequency of the system to which the machine is connected and either the number of poles or the number of projections in the machine, *torque*.

Concerning the object properties, the following exist in the ontology: *deployed on platform*, *deployed system*, *generalQuantityKind*, *has deployment*, *has subsystem*, *host*, *in deployment*, *is hosted by*, *propertyType*, *skos:inScheme*.

It uses the *code* data property.

7.3.11 Agriculture Pests ontology

The Agriculture Pests ontology is available under https://w3id.org/demeter/agri/agriPest and is briefly presented

below.

Figure 41. Visualization of the Agriculture Pests ontology

AgriPest.ttl ontology consists of the *agriPest* class. This class describes the agricultural pest. It is primarily associated with the agricultural vertical and related IoT applications. It has only one object property, the *hasAgriProductType* which is a reference to recommended types of products that can be used to treat this pest. This ontology contains the following 3 data properties: *alternateName*, *description* and *name*. Additionally, note that using the *agroVocConcept* property (from the common module), individuals of the *agriPest* class can be connected to the equivalent pest concept from AGROVOC.

7.3.12 Farm Animals ontology

The Farm Animals ontology is available under https://w3id.org/demeter/agri/farmAnimal and is presented in the figure below.

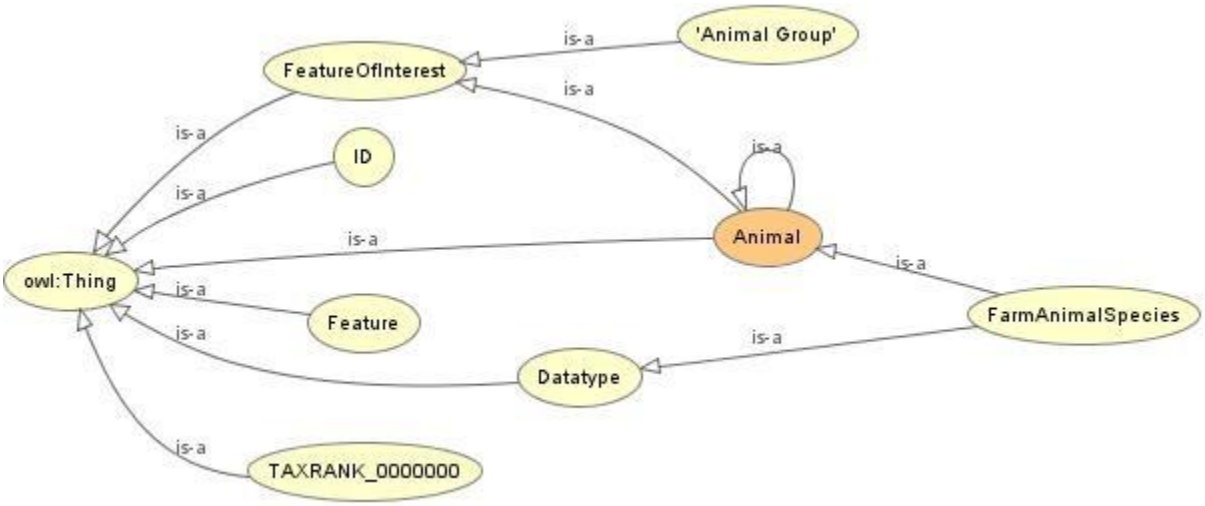


Figure 42. Visualization of the Farm Animals ontology

This module describes the proposed animal data model that has been made from a more general point of view, trying to adjust it to the information coming from the devices and sensors used to monitor or record the animals, their status, their relationships and properties in general.

The class *Datatype* encapsulates the *FarmAnimalSpecies* subclass, which is also a subclass of *Animal* and defines every possible type of animal we might encounter in our data. This subclass defines a number of restrictions that

will be explained in the next steps. The aforementioned hierarchy is expressed in *FeatureOfInterest* class which also encapsulates the *Animal Group* subclass which is a collection of animals. Other classes are *ID* and *TAXRANK_0000000* which are necessary for identifying and telling the animals apart.

Object Properties are relationships defined between class objects. Such properties could define relationships between different animals such as parenthood (*calvedBy*, *siredBy*) or between animal and person (*ownedBy*). Other relationships defined between objects of the animal subclasses are *has member* and its inverse *is member of*, *has id*, *includesAnimal*. There are also location related properties like *is located in* and its subclass *locatedAt* and the inverse *is location of*.

Likewise, Data Properties match an object to a value and not to another object. Some of these properties should be unique for each object such as *birthdate*, *has birth date*, *hasName*, *livestockNumber*, *livestockType*. Other just describe the animal like *breed*, *sex*, *weight*, *species* and most of the rest are related to the condition of the animals. These are *healthCondition*, *phenologicalCondition*, *reproductiveCondition*, *welfareCondition*. Finally, there are some other properties like *legalID* and *relatedSource*, which is the ID used for the animal in external applications.

7.4 Metadata Schema

Metadata, "a set of data that describes and gives information about other data", is a pervasive concept that is relevant to all components of an information system. There are a number of common patterns presented in the Table below, by which metadata is made available, depending on what component makes the metadata available and what type of object the metadata is related to. The AIM data model and, in particular, the cross-domain ontologies address the data item-scoped cases. This section focuses on **dataset scoped metadata** which may be delivered or referenced by several components.

Table 3. Forms, scope and examples of metadata

Form	Scope	Provided by	Example
As elements of the data itself (self-describing data)	Data item	Data item	https://www.w3.org/TR/vocab-ssn/#SOSAusedProcedure
As a reference to a metadata record embedded in the data itself	Dataset	Data item	
a metadata record embedded in data package	Dataset	Dataset	ESRI shapefile.xml

As an additional view of the data item	Data item	Data Access Service	https://citation.crosscite.org/docs.html
As description of the data provided by a data access mechanism	Dataset	Data access service instance	OGC GetCapabilities service method
As a description of the data provided as part of the description of the data access mechanism	Data access service type	Documentation	API description
As a catalog record describing a data set	Dataset	Catalog (or linked from or embedded in data item)	DCAT

The key architectural concern is for commonality of description across different implementation patterns - descriptions are hard to design, publish and interpret, so it is imperative to minimise unnecessary differences.

In order to deliver a suitable metadata schema for DEMETER, the following principles have been used to drive the respective design:

1. General principles of data modelling apply, including reuse of standards as applicable.
2. Metadata needs to be extensible, to allow sufficient detail to be included to meet different requirements in different parts of the discovery, access, processing and reporting usage chain.
3. Metadata needs to be available in the form most useful to each part of the chain, so alignments with specific metadata schema will be needed - for example, use of schema.org for discovery.
4. Where possible the metadata schema will incorporate specific metadata models needed for specific functions and data types.
5. Metadata about data can be used to inform usage through either **detailed descriptions** or **declarations of conformance**.
6. **Detailed descriptions** are limited by the expressivity of the description language, effort available to standardise descriptions of every aspect of the data, and the complexity of comparison of different descriptions to determine functional compatibility on an instance by instance basis at run-time.
7. **Declarations of conformance** require the description of conformance requirements (“conformance classes”), but these can be handled once.
8. **Declarations of conformance** and **Detailed descriptions** are complementary - conformance classes can be derived from description models, and declarations of conformance can be created by reasoning over

detailed descriptions. More generally, however conformance classes can be used to derive metadata schemas to control the detailed descriptions and to capture the relevant conformance declarations at this time.

In this framework, the challenges of comprehensive, but variable needs for detailed metadata leads to different approaches:

1. large lists of used or proposed elements covering all possibilities
2. specific minimal data common models
3. ad-hoc metadata models combining elements needed for a given task

In practice, all these patterns co-exist, and an additional pattern is introduced to provide for both simplification and extension control: **profiling.** All metadata schemas are implemented by either formal or undocumented choices. This pattern is described in the Profiles vocabulary[79] which is a cross-domain ontology specifically designed to handle the formalism of profile descriptions using an open-ended range of different possible mechanisms.

The discussion of different profiles of DCAT below highlights the nature of extensible metadata models, and the proposed use of a consistent formal profiling mechanism to integrate with DEMETER AIM.

7.4.1 Integration with DEMETER AIM

Via its comprehensive "Concern Hexagon" (cf. Figure 42), the IDS Information Model provides a solid conceptual foundation for the AIM, which is also backed by a strong technical implementation that reuses all the other metadata standards identified as relevant for DEMETER in Section 5.3 (DQV, PROV-O, etc.). However, it has not yet followed the profiling principle, which has been identified as crucial for the extensibility of the AIM above. Thus, for use in DEMETER it is not only being upgraded to DCAT 2 (the current IDS Information Model version 3.1.0 is still based on DCAT 1). Furthermore, to make sure it satisfies the requirements of DCAT-AP, it is formally defined as a DCAT profile using the Profiles vocabulary, and its relation to other existing DCAT profiles is being clarified. The resulting DCAT profile thus defines the general AIM metamodel elements to be referenced by any DEMETER generated datasets.

In particular, the DCAT-AP-DEMETER profile, which is provided by the DEMETER-adapted IDS Information Model, uses a general DCAT-DQV profile, and inherits the constraints required to use DQV with DCAT. Note DCAT-DQV itself inherits a DCAT-Datacube profile, a DCAT-PROV-O profile and a DCAT-DUV profile. While DCAT-PROV-O and DCAT-DUV are part of the meta-layer, DCAT-Datacube resides in the cross-domain layer. Hence there should be metadata profiles linked to cross-domain models to be able to provide a metadata schema optimised for each key sub-type of data. The profiles have been created by using stubs in DEMETER namespaces for each model and

[79] https://www.w3.org/TR/dx-prof/

linking those stubs as profiles without extra constraints to the official standards.

By the same mechanism as explained so far, any further AIM data model module will have a declared DCAT profile derived detailing use of the specific AIM data elements, and any inherited conformance benefits derived from the alignment of that module with standard models. Such profiles will inherit conformance claims from the cross-domain ontologies they use. For example, many AIM modules will implement a profile of sosa:Observation – so it will be possible to control metadata requirements for Observations from a single point, but also to discover all datasets that contain data using the sosa:Observation model, without having to download the data and interrogate its data type hierarchies:

a) DCAT-AP-DEMETER includes IDS constraints - and all DEMETER dataset metadatas conform to IDS, or
b) DCAT-AP-IDS is defined, and DCAT-AP-DEMETER-IDS is a subprofile to describe those datasets that conform to IDS model.

DEMETER will profile IDS where applicable and future IDS improvements, aligned with specific DEMETER needs, will facilitate the integration process.

Hence, similar to the approach taken in several recent EU initiatives such as the CYBELE project[80] , DEMETER AIM metamodel will define a profile of DCAT, which will include the relevant properties to describe datasets and other digital content relevant to the agri-food sector with focus on quality aspects, and according to IDS contraints.

[80] https://www.cybele-project.eu/

8 Semantic Interoperability Support

The state goal of the DEMETER AIM is to provide semantic interoperability with other existing systems and ontologies. Now, interoperability needs to be considered at three levels:

- **organizational level**: coordinated processes in which different organizations achieve a previously agreed and mutually beneficial goal (Policy and Behavioural);
- **semantic level**: precise meaning of exchanged information which is preserved and understood by all parties;
- **technical level**: planning of technical issues involved in linking computer systems and services (Transport and Syntactic).

Semantic interoperability allows organizations to process information from external sources in a meaningful manner. It ensures that the precise meaning of exchanged information is understood and preserved throughout exchanges between parties. **To achieve semantic interoperability, it is necessary to define the data structures and data elements for the given application domain and agree on the meaning of the information to be exchanged.** This should be the goal of any information management system.

Semantic interoperability involves semantic integration as well as, e.g., interoperability of services and tools made possible and driven by semantic integration.

Semantic integration involves the identification of logical connections (matching) between concepts in ontologies/schemas, and individuals across datasets, detecting duplicates, reconciling inconsistent data values, and reasoning with semantic mappings. For a more comprehensive overview of semantic interoperability mechanisms, refer to subsection 5.16.

In order to achieve the desired interoperability with a number of existing ontologies and systems, the DEMETER AIM has been implemented by merging, aligning and reusing terms from the most relevant ontologies and data models in the domain of agrifood and smart agriculture. In fact, what we observed is that equivalent terms for the same concepts exist in several of these. Therefore, we mostly reused (merging and aligning concepts where needed) the majority of the required terms. This will become apparent in the following sections, in which the DEMETER AIM interoperability with many dominant agri-food systems is presented. In each section, we briefly discuss the system (ontology) we aim for interoperability with and then provide the mapping (and reuse) of terms of each existing system and how these have been incorporated within the DEMETER AIM.

8.1 Semantic Mapping to FIWARE

FIWARE specifies harmonised data models that have been developed by members of the GSMA IoT Big Data

Ecosystem Project[81] and TMForum. They are work in progress being extended to cover an expanding set of IoT data. Of particular interest to our project is interoperability with the current version of the data model specific for the domain of Smart Agrifood. The current version of the model is able to cover and map a whole series of information (among the most significant) coming from the IoT sensor network: information relating to animal welfare observation, crop pest and disease management, observations regarding weather forecasts.

The data model specifies several entities/modules that are programmatically defined using a JSON Schema. The entities which are relevant to the Agrifood domain and related IoT applications are:

- *Agri App*: containing a description of a generic app made for the Agrifood domain
- *Agri Crop*: containing a description of a generic crop.
- *Agri Food*: containing a description of a generic farm made up of buildings and parcels.
- *Agri Greenhouse*: containing a description of the conditions recorded within a generic greenhouse, a type of AgriParcel.
- *Agri Parcel*: containing a description of a generic parcel of land.
- *Agri Parcel Operation*: containing a description of a generic operations performed on a parcel of land.
- *Agri Parcel Record*: containing a description of the conditions recorded on a generic parcel of land.
- *Agri Pest*: containing a description of an agricultural pest.
- *Agri Product Type*: containing a description of a generic agricultural product type. It includes a hierarchical structure that allows product types to be grouped in a flexible way.
- *Agri Soil*: containing a description of soil.
- *Animal*: containing a description of animal objects and observation of animal conditions at a certain place and time. This data model has been developed for the IoF2020 ShareBeef UC.
- *Weather Observed*: describing an observation of weather conditions at a certain place and time.
- *Weather Forecast*: containing a description of a weather forecast for a period of time and a location.
- *Weather Alert*: describing a weather alarm intended to raise attention over a forecasted extreme weather condition. This is overridden by the *Alert* module.

8.1.1 Term Mapping between FIWARE and AIM

The DEMETER AIM has been implemented by merging, aligning and reusing terms from the most relevant ontologies and data models in the domain of Agrifood and smart agriculture. One of these data models which were used is indeed the FIWARE one and its entities which were just presented. In view of this, the DEMETER AIM already includes alignments between key elements of the FIWARE data model in order to support the integration and hence interoperability with existing datasets. In particular, Table 4 provides a list of key terms from FIWARE Agrifood that, to some degree, have been re-used in and aligned with the DEMETER AIM. For each

[81] https://www.gsma.com/iot/iot-big-data/

term, we identify which FIWARE module it appears in, as well as the AIM module in which it is used, as well as the AIM mapping that enables the interoperability with it. The mapping also includes terms which are re-used in AIM and taken directly from other ontologies and data models; for those taken from FIWARE the mapping states that they are either re-used as is, or that they are re-used and aligned/merged with other dominant ontologies. In the type column, we present the type of term that is mapped and unless otherwise mentioned the mapping reference to an equivalence between the terms (e.g. equivalent class or property).

Table 4. FIWARE AgriFood term mappings to DEMETER AIM

FIWARE AgriFood term	FIWARE module	Type	AIM Module	AIM mappings
fiware:createdAt fiware:modifiedAt fiware:source fiware:dataProvider fiware:agroVocConcept	Several modules	data properties	agriCommon	fiware:createdAt fiware:modifiedAt fiware:source fiware:dataProvider fiware:agroVocConcept (re-used in AIM)
fiware:name fiware:alternateName fiware:description	Several modules	data properties	agriCommon	fiware:name fiware:alternateName fiware:description (re-used in AIM)
fiware:AgriCrop	Agri Crop	Class	agriCrop	Aligned with: saref4agri:Crop foodie:CropSpecies
fiware:cropHasAgriSoil fiware:hasAgriFertiliser fiware:hasAgriPest	Agri Crop	object properties	agriCrop	fiware:cropHasAgriSoil fiware:hasAgriFertiliser fiware:hasAgriPest (re-used in AIM)
fiware:plantingFrom fiware:harvestingInterval fiware:wateringFrequency	Agri Crop	data properties	agriCrop	fiware:plantingFrom fiware:harvestingInterval fiware:wateringFrequency (re-used in AIM)
fiware:AgriFarm	Agri Farm	Class	agriFeature	Aligned with: saref4agri:Farm inspire:Holding
fiware:AgriParcel	Agri Parcel	class	agriFeature	Aligned with: saref4agri:Parcel foodie:Plot
fiware:hasAgriCrop	Agri Parcel	Object property	agriFeature	Aligned with: foodie:crop

fiware:hasAgriParcelChildren fiware:hasAgriParcel	Agri Parcel	Object (sub)property	agriFeature	saref4agri:contains foodie: containsPlot foodie: containsZone
fiware:hasAgriParcelParent	Agri Parcel	Object (sub)property	agriFeature	saref4agri:isContainedIn
fiware:hasDevice	Agri Parcel	object property	agriFeature	fiware:hasDevice (re-used in AIM)
fiware:hasAgriSoil	Agri Parcel	object property	agriFeature	fiware:hasAgriSoil (re-used in AIM)
fiware:location fiware:landLocation	Agri Parcel	object (sub)property	agriFeature	Geo:hasGeometry
fiware:area fiware:category fiware:cropStatus fiware:lastPlantedAt	Agri Parcel	Data properties	agriFeature	fiware:area fiware:category fiware:cropStatus fiware:lastPlantedAt (re-used in AIM)
fiware:AgriParcelOperation	Agri Parcel Operation	class	agriIntervention	Aligned with: foodie:Treatment
fiware:operationHasAgriParcel	Agri Parcel Operation	Object property	agriIntervention	Aligned with: foodie: interventionPlot
fiware:hasOperator	Agri Parcel Record	Object property	agriIntervention	Aligned with: foodie:operator
fiware:hasAgriProductType	Agri Parcel Record	Object property	agriIntervention	Aligned with: foodie:treatmentProduct
fiware:plannedStartAt fiware:plannedEndAt	Agri Parcel Record	Object properties	agriIntervention	fiware:plannedStartAt fiware:plannedEndAt (re-used in AIM)
fiware:operationType	Agri Parcel Record	Object (sub)property	agriIntervention	foodie:type
fiware:result	Agri Parcel Record	Object (sub)property	agriIntervention	foodie:notes
fiware:status	Agri Parcel Record	Object property	agriIntervention	Aligned with: foodie: status
fiware:startedAt fiware:endedAt	Agri Parcel Record	Data properties	agriIntervention	Aligned with: (FOODIE) baseInspire:validFrom baseInspire:validTo
fiware:quantity	Agri Parcel	Data	agriIntervention	Aligned with:

	Record	properties	n	foodie: quantity
fiware:reportedAt	Agri Parcel Record	Data properties	agriIntervention	fiware:reportedAt (re-used in AIM)
fiware:waterSource	Agri Parcel Record	Data properties	agriIntervention	fiware:waterSource (re-used in AIM)
fiware:Alert	Alert	Class	agriAlert	Aligned with: foodie: Alert
fiware:validFrom, validTo, dateIssued	Alert	data (sub)property	agriAlert	foodie:alertDate
fiware:location	Alert	data (sub)property	agriAlert	foodie:alertGeometry
fiware:category fiware:subCategory fiware:address fiware:alertSource fiware:data fiware:severity	Alert	data properties	agriAlert	fiware:category fiware:subCategory fiware:address fiware:alertSource fiware:data fiware:severity (re-used in AIM)
fiware:AgriProductType	AgriProduct Type	class	agriProduct	Aligned with: foodie: Product
fiware:name	AgriProduct Type	Included in data properties	agriProduct	foodie:productName foodie:nutrientName foodie:ingredientName
fiware:hasAgriProductTypeParent fiware:hasAgriProductTypeChildren	AgriProduct Type	Object properties	agriProduct	fiware:hasAgriProductTypeParent fiware:hasAgriProductTypeChildren (re-used in AIM)
fiware:weatherType fiware:airTemperature fiware:soilTemperature fiware:soilMoistureVwc fiware:soilMoistureEc fiware:solarRadiation fiware:atmosphericPressure fiware:dewPoint fiware:visibility fiware:temperature fiware:relativeHumidity	Agri Parcel Record (some from Weather observed module)	data (sub)properties	agriProperty	Defined and aligned as subproperties of: saref:hasValue

fiware:precipitation fiware:windDirection fiware:windSpeed fiware:atmosphericPressure fiware:pressureTendency fiware:solarRadiation fiware:illuminance fiware:streamGauge fiware:snowHeight				
fiware:recordHasAgriParcel fiware:hasDevice fiware:refDevice fiware:refPointOfInterest	Agri Parcel Record /Weather observed	Object properties	agriProperty	fiware:recordHasAgriParcel fiware:hasDevice fiware:refDevice fiware:refPointOfInterest (re-used in AIM)
fiware:AgriParcelRecord	Agri Parcel Record	class	agriProperty	fiware:AgriParcelRecord (re-used in AIM)
fiware:WeatherObserved	Weather Observed	class	agriProperty	fiware:WeatherObserved (re-used in AIM)
fiware:WeatherForecast	Weather Forecast	class	agriProperty	fiware:WeatherForecast (re-used in AIM)
fiware:observedAt fiware:dateObserved	Agri Parcel Record /Weather observed	Data properties	agriProperty	Aligned with: saref:hasTimestamp
fiware:AgriPest	Agri Pest	class	agriPest	fiware:AgriPest (re-used in AIM)
fiware:hasAgriProductType	Agri Pest	Object property	agriPest	fiware:hasAgriProductType (re-used in AIM)
fiware:Animal	Animal	class	farmAnimal	saref4agri:Animal inspire: FarmAnimalSpecies (subclass)
fiware:species	Animal	Data property	farmAnimal	Aligned with: foodie: livestockType
fiware:legalID	Animal	property	farmAnimal	Aligned with: foodie: livestockNumber
fiware:calvedBy fiware:siredBy fiware:ownedBy fiware:locatedAt	Animal	Object properties	farmAnimal	All these are re-used in AIM

fiware:fedWith				
fiware:legalID fiware:sex fiware:birthdate fiware:relatedSource fiware:breed fiware:weight fiware:phenologicalCondition fiware:reproductiveCondition fiware:healthCondition fiware:welfareCondition	Animal	Data properties	farmAnimal	All these are re-used in AIM

From the table above it is obvious that there is extensive reuse of FIWARE terms and concepts within AIM, which is intentional in order to promote extensive mapping and interoperability between AIM and FIWARE. At this point, what is not covered are essentially the Agri App and the Agri Greenhouse modules of FIWARE together with some parts of Agri Soil (although some elements of the latter appear within the agri parcel and the agri product mappings).

8.1.2 The NGSI-LD connection

Another point of interoperability between the DEMETER AIM and FIWARE in general is the fact that DEMETER has selected to use the NGSI-LD representation as the core meta-model; this will of course be used when implementing the DEMETER enabled apps. Remember that NGSI-LD is an evolution of the NGSI context interface family, particularly the FIWARE NGSI v2 information model, which was evolved by ETSI ISG CIM initiative to support linked data, property graphs and semantics. It focuses on the management of context information, which facilitate the development of smart solutions for different domains including smart agrifood. Here, context comprises all characteristics of all the entities (physical and nonphysical) involved in a target system/environment, as well as their states and other dynamic properties, together with relationships that stand for actual and virtual connections between them [ETS6].

Furthermore, the NGSI-LD meta-model provides a formal basis for representing "property graphs" using RDF/RDFS/OWL, making it possible to perform back and forth conversion between datasets based on the property graph model and linked data datasets that rely on RDF using blank-nodes reification, which will of course, be very useful for enabling semantic interoperability with all the systems that DEMETER has semantic mappings (and which are presented in the current section). This could be described as raising the semantic expressivity of RDF triples to the level of property graphs, as for instance, property graphs may use predicates as subjects of other predicates (properties of properties and properties of relationships). Conversely, it may be described as grounding the semantics of property graph elements in discoverable definitions and using this to

constrain arbitrary and non-interoperable proliferation of similar property graph patterns for the many specific cases that need to be modelled.

Thus overall, the use of NGSI-LD allows achieving the semantic referencing, where elements in the AIM model can be matched to entities in these well-known ontologies and systems such as FIWARE, but also the rest we present later in this section.

8.2 Semantic Mapping to Saref4Agri

Saref4Agri is one of the data models on which the DEMETER AIM is based on (and is presented in detail in Section 5.5 of this document). Most of the classes, properties and individuals in saref4agri ontology are reused in the AIM domain-specific ontologies. In view of this, the DEMETER AIM already includes alignments between key elements of the Saref4Agri model in order to support the integration and hence interoperability with existing datasets.

8.2.1 Data Elements Mapping

The following table provides a list of key terms from Saref4Agri that, to some degree, have been re-used in and aligned with the DEMETER AIM. For each term, we identify the AIM module in which it is used, as well as the AIM mapping that enables the interoperability with it.

Table 5. Mapping of Saref4Agri OM Classes to AIM

Saref4Agri term	Type	AIM Module	AIM mappings
saref4agri:hasName	Data property	agriCommon agriFeature farmAnimal	saref4agri:hasName (re-used in AIM) Aligned with: fiware:name
saref:hasDescription	Data property	agriCommon	saref:hasDescription (re-used in AIM) Aligned with: foodie:description,fiware:description
saref4agri:managesFarm	Object property	agriCommon	saref4agri:managesFarm (re-used in AIM)
saref4agri:Farm saref4agri:FarmHolding saref4agri:Farmer	Class	agriCommon	saref4agri:Farm saref4agri:FarmHolding saref4agri:Farmer (re-used in AIM)

saref4agri:Farm	Class	agriFeature	saref4agri:Farm (re-used in AIM) Aligned with: af-inspire:Holding, fiware:AgriFarm
saref4agri:Parcel	Class	agriFeature	saref4agri:Parcel (re-used in AIM) Aligned with: foodie:Plot, fiware:AgriParcel
saref4agri:Building saref4agri:BuildingSpace saref4agri:Crop	Class	agriFeature	saref4agri:Building saref4agri:BuildingSpace saref4agri:Crop (re-used in AIM)
saref4agri:contains	Object property	agriFeature	saref4agri:contains (re-used in AIM) Parent property of: af-inspire:contains, foodie:containsPlot, foodie:containsZone, fiware:hasAgriParcelChildren, fiware:hasAgriParcel
saref4agri:isContainedIn	Object property	agriFeature	saref4agri:isContainedIn (re-used in AIM) Parent property of: foodie:holdingSite, foodie:holdingPlot, foodie:holdingZone, fiware:hasAgriParcelParent
saref4agri:Crop	Class	agriCrop	saref4agri:Crop (re-used in AIM) Aligned with: foodie:CropSpecies, fiware:AgriCrop
saref4agri:hasHarvestDate	Data property	agriCrop	saref4agri:hasHarvestDate (re-used in AIM) Aligned with: foodie:productionDate
saref4agri:hasPlantDate	Data property	agriCrop	saref4agri:hasPlantDate (re-used in AIM)
saref4agri:generates saref4agri:receives	Object property	agriCrop	saref4agri:generates, saref4agri:receives (re-used in AIM)
saref4agri:Soil	Class	agriProperty	saref4agri:Soil

			(re-used in AIM)
saref4agri:AirTemperature saref4agri:AmbientHumidity saref4agri:IrrigationWater saref4agri:PlantGrowthStage saref4agri:Precipitation saref4agri:SoilMoisture	Individual	agriProperty	saref4agri:AirTemperature saref4agri:AmbientHumidity saref4agri:IrrigationWater saref4agri:PlantGrowthStage saref4agri:Precipitation saref4agri:SoilMoisture (re-used in AIM)
saref4agri:SoilTemperature	Individual	agriProperty	saref4agri:SoilTemperature (re-used in AIM) Aligned with: cf:soil_temperature
saref4agri:SoilTensiometer saref4agri:Thermometer saref4agri:WateringGun saref4agri:WateringSystem saref4agri:WateringValve saref4agri:WeatherStation saref4agri:WeightSensor saref4agri:Pluviometer saref4agri:EatingActivitySensor saref4agri:MilkingSensor saref4agri:MovementActivitySensor	Class	agriSystem	saref4agri:SoilTensiometer saref4agri:Thermometer saref4agri:WateringGun saref4agri:WateringSystem saref4agri:WateringValve saref4agri:WeatherStation saref4agri:WeightSensor saref4agri:Pluviometer saref4agri:EatingActivitySensor saref4agri:MilkingSensor saref4agri:MovementActivitySensor (re-used in AIM)
saref4agri:Animal	Class	farmAnimal	saref4agri:Animal (re-used in AIM) Aligned with: fiware:Animal Parent class of: af-inspire:FarmAnimalSpecies
saref4agri:AnimalGroup saref4agri:ID	Class	farmAnimal	saref4agri:AnimalGroup saref4agri:ID (re-used in AIM)
saref4agri:hasBirthDate	Data property	farmAnimal	saref4agri:hasBirthDate (re-used in AIM) Aligned with: fiware:birthdate
saref4agri:hasDeathDate	Data	farmAnimal	saref4agri:hasDeathDate

	property		(re-used in AIM)
saref4agri:isLocatedIn	Object property	farmAnimal	saref4agri:isLocatedIn (re-used in AIM) Parent property of: fiware:locatedAt
saref4agri:isLocationOf saref4agri:isMemberOf saref4agri:hasID saref4agri:hasMember	Object property	farmAnimal	saref4agri:isLocationOf saref4agri:isMemberOf saref4agri:hasID saref4agri:hasMember (re-used in AIM)
saref4agri:Platform	Class	agriSystem	saref4agri:Platform (re-used in AIM) Parent property of: Fiware*:PhysicalObject ssn:Platform
saref4agri:Property	Class	agriCrop agriProperty	ssn:property
saref4agri:Humidity	Class	agriProperty	saref:humidity
saref4agri:Soil moisture	Class	agriSystem	saref4agri:soil moisture (re-used in AIM)
saref4agri:Intake saref4agri:Irrigation water saref4agri:Plant growth stage saref4agri:Precipitation saref4agri:Temperature saref4agri:Air temperature saref4agri:Soil temperature saref4agri:Time saref4agri:Yield	Class	agriProperty	saref4agri:humidity saref4agri:temperature saref:temperature (re-used in AIM) Parent property of: Fiware*:humidity Fiware*:temperature ssn:time
saref4agri:Temporal entity	Data	agriProperty	saref4agri:Temporal entity

saref4agri:instant saref4agri:time interval	Property	agriAlert agriCommon agriIntervention	saref:Property (re-used in AIM) Parent property of: Fiware: Individuals foodie:Measure ssn:time
saref4agri:Unit of measure	Class	agriProperty	saref4agri:Unit of measure saref:Unit of measure (re-used in AIM) Parent property of: Fiware*:Unit foodie:Unit of measure

8.2.2 Semantic Relationships

The Table below describes the main properties of Saref4Agri ontology.

Table 6. Main classes and properties of the Saref4Agri ontology

Class	Property	Description
s4agri:Deployment	ssn:deployedOnPlatform some sosa:Platform	The relation between a deployment and the platform in which it is deployed.
	ssn:deployedSystem some ssn:System	The relation between a deployment and the system deployed.
	s4agri:hasDeployementPeriod some time:TemporalEntity	The relation between a deployment and the time span during which the systems are deployed.
	s4agri:isDeployedAtSpace somegeosp:SpatialObject	The relation between a deployment and the spatial area in which the systems are deployed.
s4agri:Animal	s4agri:hasBirthDate max 1 xsd:dateTime	The birth date of an animal.
	s4agri:hasDeathDate max 1 xsd:dateTime	The death date of an animal.
	s4agri:hasID exactly 1 s4agri:ID	The unique identifier of an animal.
	s4agri:isLocatedIn some geo:Feature	The physical location of an animal
	s4agri:isMemberOf some s4agri:AnimalGroup	An animal can be part of groups.
	s4agri:name max 1 xsd:string	The name of an animal

s4agri:AnimalGroup	s4agri:hasMember some s4agri:Animal	The members of an AnimalGroup.
	s4agri:receives some s4agri:Intake	The intake/consumption of an AnimalGroup.
	s4agri:generates some s4agri:Yield	The yield generated by an AnimalGroup.
	s4agri:isLocatedIn some geo:Feature	The physical location of an AnimalGroup.
	s4agri:name max 1 xsd:string	The name of an AnimalGroup.
s4agri:Crop	s4agri:receives some s4agri:Intake	The intake/consumption of certain substances in a Crop.
	s4agri:generates some s4agri:Yield	The yield generated by a Crop.
	s4agri:hasPlantDate some xsd:DateTime	The day the crop is planted.
	s4agri:hasHarvestDate some xsd:DateTime	The day the crop is harvested.
s4agri:Parcel	s4agri:contains some s4agri:Crop	A parcel can contain some crops.
	s4agri:name max 1 xsd:string	The name of a parcel

8.3 Semantic Mapping to ADAPT

The Agricultural Data Application Programming Toolkit (ADAPT)[82] was created by the non-profit consortium AgGateway[83], which is dedicated to the implementation of standards to advance digital agriculture. In the scope of ADAPT, also the Common Object Model for field operations as well as a set of format conversion tools were created. The ADAPT creator's mission is to create a framework which facilitates interoperability in the agricultural sector, particularly regarding communication between growers, machines and their partners. The suitability for interoperability with DEMETER AIM will be assessed the following. Further details of ADAPT are provided in Section 5.13.

The ADAPT framework incorporates a "superset" of data models used in the agriculture and has a larger scope than ISO 11783[84] but is interoperable with it at the same time. ISO11783 consists of two part, one being commonly referred to as ISOBUS (machine control) and the other one ISOXML (farm management). The ADAPT extension facilitates integration with non-ISOXML machines, offers an improved resource identification system and takes contextual information into account. However, the scope of the ADAPT object model is specific to farming applications and does not cover, e.g., the animal husbandry aspect covered by AIM. Interoperability between ADAPT and AIM can hence only be expected for an AIM-subset.

[82] https://aggateway.atlassian.net/wiki/spaces/ADM/pages/53248025/ADAPT+Common+Object+Model+Documentation
[83] www.aggateway.org
[84] https://aggateway.atlassian.net/wiki/spaces/ADM/pages/165942474/ADAPT+Frequently-Asked+Questions+FAQ

8.3.1. Data Elements Mapping

A comprehensive sample of relevant ADAPT Object Model classes and potential mappings to DEMETER AIM classes are shown in Table 7.

Table 7. Mapping of ADAPT OM Classes to AIM

ADAPT Object Model Class	Mapped to these classes via AIM
Crop / CropVariety / Trait	FOODIE cropSpecies and cropType FIWARE AgriCrop Agrifood Saref4agri s4agri:Crop and s4agri:PlantGrowthStage
CropProtectionProduct	FIWARE AgriPest, Foodie Pest
CropZone, Field, FieldBoandary	FIWARE AgriFarm, AgriParcel FOODIE Farm, Plot and Management Zone
CropNutritionIngedient, CropNutritionProduct	FOODIE Fertilization
DataQualityElement	dqv:Metric
DataQualityMeasure	dqv:QualityMeasurement
DataQualityEvaluationMethod	dqv:QualityAnnotation
UnitOfMeasure	qudt:Unit
UnitOfMeaureDimensionEnum	qudt:hasDimension
NumericRepresentation	qudt:numericValue
DateTime	time:DateTimeDescription
Obs	sosa:Observation
Observations	sosa:Observation
OMCode	sosa:ObservableProperty
Location	geo:asWKT/asGML
BoundingBox	geo:Geometry
Shape/ShapeTypeEnum	geo:Geometry

As Table 7 indicates, there is an overlap between concepts in DEMETER AIM and concepts in the ADAPT model.

When it comes to cross-domain / meta concepts, such as time, geo, sensor observation and data quality, AIM utilized existing standards, whereas ADAPT builds upon a proprietary class system. Domain-specific concepts such as crop types, fertilization, machinery and fields are also overlapping with DEMETER AIM-adopted standards such as FOODIE, FIWARE and Saref4Agri. However, a certain level of interoperability with ADAPT is given by the fact that FIWARE utilizes the richness of the ADAPT Object Model in the area of machinery by using ADAPT object model concepts within the FIWARE model:

Table 8. Excerpt of the FIWARE Device Data Model Documentation

- `implement`: A device used or needed in a given activity; tool, instrument, utensil, etc. https://github.com/ADAPT/ADAPT/blob/develop/source/ADAPT/Equipment/ImplementConfiguration.cs
- `irrSystem`: A mobile or fixed irrigation system such as a center pivot, linear, traveling gun, solid set, etc. https://github.com/ADAPT/ADAPT/blob/develop/source/ADAPT/Equipment/IrrSystemConfiguration.cs
- `irrSection`: A section of an IrrSystem. Different enough from a regular section. https://github.com/ADAPT/ADAPT/blob/develop/source/ADAPT/Equipment/IrrSectionConfiguration.cs
- `endgun`: A device attached to an irrigation system that projects water beyond it https://github.com/ADAPT/ADAPT/blob/develop/source/ADAPT/Equipment/EndgunConfiguration.cs

The ADAPT object model does not contain any domain-specific concepts regarding animal husbandry, hence ADAPT only covers a subset of DEMETER AIM domains. On the other hand, it offers a deep level of detail in the machinery data domain. Mapping of those concepts to DEMETER AIM would likely lead to a certain degree of semantic information loss.

The concept “Compound Identifiers” appears to be integral to the ADAPT object model and it is still up to decide whether a similar concept is to be adopted within DEMETER AIM. If so, and if it would be based on ADAPT-conform identifiers, it would largely increase the feasibility for an interoperability between ADAPT and DEMETER AIM. If interoperability was to be established, another hurdle would be the alignment of, for example, the classes listed in Table 5. To maintain the functionality of the ADAPT Framework, a cautious mapping from AIM concepts to ADAPT concepts and vice versa would have to be established.

As interoperability mechanism, ADAPT offers a plugin interface. This is rather targeted at data providers, such as machinery producers, and offers them a way to establish interoperability with the ADAPT data space. The ADAPT documentation adds following to it: “Converting formats, however, is not enough to guarantee interoperability; a system of shared meaning is also required. For this reason, ADAPT was designed with an emphasis on unique identifiers, and the use of data-type registries and other semantic assets that can ensure that all participants in a data exchange process interpret the data in the same way. The end-goal is the OEMs and software developers will develop plug-ins to exchange data from their systems to/from the ADAPT data format – providing for the free exchange of logged agronomic data, allowing for the growth of the use of Field Operations Data in

Agriculture as Growers will be able to use the hardware and software of their choice to run their businesses without worry of the systems not being interoperable."[85] Even though and integration with other "interoperability spaces" (such as DEMETER) is not officially a use case of ADAPT Plug-ins, they might still be the best starting point for integration/mapping efforts.

For further support considering plug-in implementation, which might be the most viable alleyway towards establishing interoperability with AIM, the ADAPT team suggests to join the ADAPT Technical Team Meetings via adapt.feedback@aggateway.org and provides further explanations in the form of a video tutorial at https://www.youtube.com/watch?v=MY63gGSECoI ("mechanics of plugin manager")[3].

8.4 Semantic Mapping to INSPIRE and FOODIE

The INSPIRE directive[86] aims at building a Pan-European spatial data infrastructure (SDI), requiring the EU Member States to make available spatial data, from multiple thematic areas, according to established implementing rules using appropriate services (as described in the subsequent subsections). The INSPIRE Implementing Rules (IRs) and Technical Guidelines (Data Specifications)[87] specify common data models, code lists, map layers and additional metadata on the interoperability to be used when exchanging spatial datasets. The data specifications are based on ISO/OGC standards for geospatial services and formats[88], thus applying the ISO/OGC-approach of modelling physical things, so-called "features". INSPIRE covers 34 spatial data themes, being the most relevant for DEMETER the Agricultural and Aquaculture Facilities (hereinafter AF) [AF13] that defines a model composed of core information in relation to the geographical description of entities under the Agriculture and Aquaculture scope. In particular, the INSPIRE directive defines Agricultural and aquaculture facilities as "farming equipment and production facilities (including irrigation systems, greenhouses and stables)". The AF data model itself is based on the Activity Complex model [AC13]. "Activity Complex" is in INSPIRE a generic name agreed across thematic domains trying to avoid specific thematic connotations such as "Plant", "Installation", "Facility", "Establishment" or "Holding. In AF data model, the Activity Complex model is extended to the basic Agricultural and Aquaculture features Holding and Site. These features contain only basic information about the location of the Holding and the Site, the type of activities performed on that locations, and just in case that animals are kept, what type of animals is kept on the Site. In particular, Holding is regarded as a specialisation of an Activity Complex, and it contains at least one or more Sites, which can keep none, one or more animal species. The AF specification also includes an extended model to represent complementary information about Agricultural and Aquaculture Facilities. The latest specification release (v3.0), includes extensions about plots, agri-buildings, installations, irrigation and drainage, farm animals and animal health. A

[85] https://aggateway.atlassian.net/wiki/spaces/ADM/pages/165942474/ADAPT+Frequently-Asked+Questions+FAQ
[86] http://inspire.ec.europa.eu/
[87] https://inspire.ec.europa.eu/data-specifications/2892
[88] http://www.opengeospatial.org/standards/is

detailed information about these elements is provided in Section 5.6.

FOODIE model, and related ontology, is an extension of the AF model. It is worth noting, though, that FOODIE was built based on the original core AF data specification, and thus a main motivation was to represent a continuous area of agricultural land with one type of crop species, cultivated by one user in one farming mode, which was missing in that AF specification. Such feature was called Plot and was providing a more detailed level than Site that was already part of the AF data model. Additionally, FOODIE includes many different concepts to represent agro-related information, such as ProductionType (for representing production-related data), CropSpecies (for representing the planted crop species), Intervention and Treatment, and even Alert to represent alerts generated by the models integrated in a platform. A detailed description of FOODIE model is provided in Section 5.10.

8.4.1 INSPIRE and FOODIE Interoperability

INSPIRE data specifications provide the basis for the interoperability of spatial data sets and services across member states. In particular, interoperability in INSPIRE means the possibility to combine spatial data and services from different sources across the European Community in a consistent way without involving specific efforts of humans or machines. In this line, interoperability may be achieved by either changing (harmonising) and storing existing data sets or by transforming them via services for publication in the INSPIRE infrastructure. Hence, it would be expected that users will spend less time and efforts on understanding and integrating data when they build their applications based on data delivered in accordance with INSPIRE.

FOODIE is compatible and extends INSPIRE AF data model, and so it carries on with a similar goal as INSPIRE but focused and extended on the Agriculture domain. By providing an application vocabulary covering different categories of information dealt by typical farm management tools/apps, FOODIE aimed at enabling their interoperability and to be compliant with INSPIRE directive and ISO standards.

8.4.2 INSPIRE and FOODIE mappings

DEMETER AIM has been implemented by reusing terms from the most relevant ontologies and data models in the different areas relevant to support the final DEMETER applications, and using as a base the NGSI-LD meta-model and approach. Hence, DEMETER AIM defines for each (sub-)domain a set of modules facilitating the scalability and maintainability of the model. Additionally, DEMETER AIM already includes alignments between key elements of these models in order to support the integration of existing datasets (which are based on them). In particular, regarding the agrifood domain, DEMETER AIM defines over 10 modules reusing terms from SAREF4Agri, FIWARE Agrifood data models, and FOODIE (and thus INSPIRE AF core data model). The following table provides a list of key terms from FOODIE/INSPIRE that have been re-used in DEMETER AIM and how they have been mapped to terms in other ontologies/models (if applicable). Additionally, Table 9 also shows terms from INSPIRE AF extended model, which are not yet covered directly in DEMETER AIM, with the applicable

mappings.

Table 9. INSPIRE and FOODIE mappings (mappings with * are not yet in DEMETER AIM)

INSPIRE/FOODIE term	type	AIM Module	AIM mappings
Holding (INSPIRE)	class	agriFeature	saref4agri:Farm fiware:AgriFarm
Site (INSPIRE)	class	agriFeature	foodie:Site (re-used in AIM)
Plot (FOODIE) – equivalent to INSPIRE AF extended model Plot*	class	agriFeature	saref4agri:Parcel fiware:AgriParcel
ManagementZone (FOODIE)	class	agriFeature	foodie:ManagementZone (re-used in AIM)
AgriBuilding (INSPIRE AF extended model)	class	agriFeature	saref4agri:Building*
crop	property	agriFeature	fiware:hasAgriCrop
contains (INSPIRE), containsPlot (FOODIE), containsZone (FOODIE)	property	agriFeature	saref4agri:contains fiware:hasAgriParcelChildren (subproperty)
holdingSite (FOODIE), holdingPlot (FOODIE), holdingZone (FOODIE)	property	agriFeature	saref4agri:isContainedIn fiware:hasAgriParcelParent (subproperty)
Treatment (FOODIE)	class	agriIntervention	fiware:AgriParcelOperation
interventionPlot (FOODIE)	property	agriIntervention	fiware:operationHasAgriParcel
type (FOODIE)	property	agriIntervention	fiware:operationType (subproperty)
status (FOODIE)	property	agriIntervention	fiware:status
operator (FOODIE)	property	agriIntervention	fiware:hasOperator
validFrom, validTo (FOODIE)	property	agriIntervention	fiware:startedAt, fiware:endedAt
treatmentProduct (FOODIE)	property	agriIntervention	fiware:hasAgriProductType
quantity (FOODIE)	property	agriIntervention	fiware:quantity
CropSpecies (FOODIE)	class	agriCrop	saref4agri:Crop fiware:AgriCrop

CropType (FOODIE)	class	agriCrop	foodie:CropType (re-used in AIM)
ProductionType (FOODIE)	class	agriCrop	foodie:ProductionType (re-used in AIM)
productionDate (FOODIE)	property	agriCrop	saref4agri:hasHarvestDate
Product (FOODIE)	class	agriProduct	fiware:AgriProductType
PropertyType (FOODIE)	class	agriProperty	saref:Property
productionProperty, soilProperty (FOODIE)	property	agriProperty	saref:hasProperty
quantitativeProperty, productionAmount (FOODIE)	property	agriProperty	saref:relatesToMeasurement
organicMatter, electricConductivity, soilType, soilTexture, pH (FOODIE)	Individual	agriProperty	saref:Property (type)
MachineType, TractorType (FOODIE)	class	agriSystem	sosa:Platform (used by Saref)
Installation, AquacultureInstallation, WaterManagementInstallation (INSPIRE AF extended model)	class	agriSystem	saref4agri:Deployment*
FarmAnimalSpecies (INSPIRE)	class	farmAnimal	saref4agri:Animal (subclass) fiware:Animal (subclass)
livestockType (FOODIE)	property	farmAnimal	fiware:species
livestockNumber (FOODIE)	property	farmAnimal	fiware:legalID
RecognisedHealthStatus (INSPIRE AF extended model)	class	farmAnimal	*
Alert (FOODIE)	class	agriAlert	fiware:Alert
alertDate (FOODIE)	property	agriAlert	fiware:validFrom (subproperty)
alertGeometry (FOODIE)	property	agriAlert	fiware:location
description (FOODIE)	property	agriCommon	saref:hasDescription fiware:description
generatedAtTime (FOODIE)	property	agriCommon	saref:hasTimestamp fiware:createdAt
ResponsibleParty (INSPIRE)	class	agriCommon	foaf:Agent (subclass)
code (FOODIE)	property	agriCommon	foodie:code (re-used in AIM)

notes (FOODIE)	property	agriCommon	foodie:notes (re-used in AIM)
alertPlot (FOODIE)	property	agriAlert	foodie:alertPlot (re-used in AIM)
alertZone (FOODIE)	property	agriAlert	foodie:alertZone (re-used in AIM)
alertSpecies (FOODIE)	property	agriAlert	foodie:alertSpecies (re-used in AIM)
plotAlert (FOODIE)	property	agriAlert	foodie:plotAlert (re-used in AIM)
zoneAlert (FOODIE)	property	agriAlert	foodie:zoneAlert (re-used in AIM)
speciesAlert (FOODIE)	property	agriAlert	foodie:speciesAlert (re-used in AIM)
originType	property	agriFeature	foodie:originType (re-used in AIM)
originTypeValue	individual	agriFeature	foodie:originTypeValue (re-used in AIM)
variety	property	agriCrop	foodie:variety (re-used in AIM)
family	property	agriCrop	foodie:family (re-used in AIM)
species	property	agriCrop	foodie:species (re-used in AIM)
genus	property	agriCrop	foodie:genus (re-used in AIM)
propertyName	property	agriProperty	foodie:propertyName (re-used in AIM)
nonQuantitativeProperty	property	agriProperty	foodie:nonQuantitativeProperty (re-used in AIM)
analysisDate	property	agriProperty	foodie:analysisDate (re-used in AIM)
manufacturer	property	agriProduct	foodie:manufacturer (re-used in AIM)
nutrient	property	agriProduct	foodie:nutrient

			(re-used in AIM)
solventQuantity	property	agriProduct	foodie:solventQuantity (re-used in AIM)
safetyPeriod	property	agriProduct	foodie:safetyPeriod (re-used in AIM)
productQuantity	property	agriProduct	foodie:productQuantity (re-used in AIM)
ingredientAmount	property	agriProduct	foodie:ingredientAmount (re-used in AIM)
nutrientAmount	property	agriProduct	foodie:nutrientAmount (re-used in AIM)
price	property	agriProduct	foodie:price (re-used in AIM)
productSubType	property	agriProduct	foodie:productSubType (re-used in AIM)
storageHandling	property	agriProduct	Foodie:storageHandling (re-used in AIM)
productName	property	agriProduct	foodie:productName (re-used in AIM)
productCode	property	agriProduct	foodie:productCode (re-used in AIM)
registrationCode	property	agriProduct	foodie:registrationCode (re-used in AIM)
ingredientName	property	agriProduct	foodie:ingredientName (re-used in AIM)
nutrientName	property	agriProduct	foodie:nutrientName (re-used in AIM)
nutrientMeasure	property	agriProduct	foodie:nutrientMeasure (re-used in AIM)
ProductPreparation	class	agriProduct	foodie:ProductPreparation (re-used in AIM)
ActiveIngredients	class	agriProduct	foodie:ActiveIngredients (re-used in AIM)
ProductNutrients	class	agriProduct	foodie:ProductNutrients (re-used in AIM)

ProductKindValue	class	agriProduct	foodie:ProductKindValue (re-used in AIM)
interventionZone	property	agriIntervention	foodie:interventionZone (re-used in AIM)
interventionGeometry	property	agriIntervention	foodie:interventionGeometry (re-used in AIM)
supervisor	property	agriIntervention	foodie:supervisor (re-used in AIM)
evidenceParty	property	agriIntervention	foodie:evidenceParty (re-used in AIM)
formOfTreatment	property	agriIntervention	foodie:formOfTreatment (re-used in AIM)
areaDose	property	agriIntervention	foodie:areaDose (re-used in AIM)
plan	property	agriIntervention	foodie:plan (re-used in AIM)
motionSpeed	property	agriIntervention	foodie:motionSpeed (re-used in AIM)
quantity	property	agriIntervention	foodie:quantity (re-used in AIM)
flowAdjustment	property	agriIntervention	foodie:flowAdjustment (re-used in AIM)
applicationWidth	property	agriIntervention	foodie:applicationWidth (re-used in AIM)
pressure	property	agriIntervention	foodie:pressure (re-used in AIM)
planProduct	property	agriIntervention	foodie:planProduct (re-used in AIM)
campaign	property	agriIntervention	foodie:campaign (re-used in AIM)
minimumDose	property	agriIntervention	foodie:minimumDose (re-used in AIM)
maximumDose	property	agriIntervention	foodie:maximumDose (re-used in AIM)
period	property	agriIntervention	foodie:period

			(re-used in AIM)
creationDateTime	property	agriIntervention	foodie:creationDateTime (re-used in AIM)
treatmentDescription	property	agriIntervention	foodie:treatmentDescription (re-used in AIM)
treatmentPlanCode	property	agriIntervention	foodie:treatmentPlanCode (re-used in AIM)
treatmentPlanCreation	property	agriIntervention	foodie:treatmentPlanCreation (re-used in AIM)
TreatmentPlan	class	agriIntervention	foodie:TreatmentPlan (re-used in AIM)
TreatmentPurposeValue	class	agriIntervention	foodie:TreatmentPurposeValue (re-used in AIM)
TreatmentValue	class	agriIntervention	foodie:TreatmentValue (re-used in AIM)
DoseUnit	class	agriIntervention	foodie:DoseUnit (re-used in AIM)
CampaignType	class	agriIntervention	foodie:CampaignType (re-used in AIM)

8.5 Semantic Mapping to AGROVOC

Information Systems can model data/information/knowledge using several approaches, but most of them are based on JSON[89] or XML[90] formats, especially in combination with non-relational databases technologies. These formats are often used to implement well-known standard schemas to facilitate the development of end-user applications. The understanding of the data itself, when similar schema elements can take on different but closely related meanings, is only partially supported by knowing the semantics of these schemas. Data correlation very often coincides with the possibility to build links between different data formats and schemas. These relationships may arise from the need to correlate this data with vocabularies such as Agrovoc, developed by FAO[91] for all terms relating to agriculture (e.g. for agricultural products description).

The DEMETER Platform will base its data model (AIM) on standard cross-domain models including SKOS, which

[89] https://www.json.org/json-en.html
[90] https://www.w3.org/XML/
[91] http://www.fao.org/home/en/

is the basis for how Agrovoc is encoded.[92] Now, since the implementation of DEMETER AIM is based on the NGSI-LD standard, this means that semantic interoperability with Agrovoc is as simple as using Agrovoc terminology references within the JSON-LD format as described below. A more indirect approach can coexist, whereby other terminology used within DEMETER can be cross-referenced to Agrovoc equivalents as required. The interoperability of the implementation with the Agrovoc infrastructure will require negotiation, since the Agrovoc services use its own pre-standard approaches, however the underlying semantic interoperability concerns are readily dealt with using the ontological meta-model for the DEMETER AIM.

The *Agrovoc vocabulary* is a collection of component vocabularies related to the Agrifood, developed to support semantic representations and data modelling. The use of such standard vocabulary can ensure both interoperability and absence of ambiguity in the data interpretation process. Agrovoc is today the most complete multilingual controlled vocabulary for agriculture. One of the predominant aspects, which characterizes it more than other vocabularies/thesauruses is represented by its multilingual character, also and especially for those applications that involve interactions with multiple users (e.g. Web Platform). Even if today it is possible to encode metadata from several languages into English (which still represents the predominant language for ontologies and vocabularies), in the Agrifood sector it is essential to be able to have access to different countries language labels (e.g. Czech, Danish, German, Italian, Polish, Portuguese, Slovak and Thai.) for different concepts.

Originally, Agrovoc was designed for indexing literature, but it is also increasingly used to facilitate the sharing and exchange of knowledge through electronic media and data formats. In addition, it contains over 40,000 concepts in a maximum of 21 languages they cover a whole range of topics related to the agriculture sector such as food, nutrition, fishing, forestry, environment and other related sectors. The vocabulary looks like an RDF[93] using the SKOS[94] standard (i.e. the de-facto standard for sharing and linking knowledge organization systems as Linked Data[95]) represented as an Agrovoc model, while all concepts are identified by URL[96]. Figure 43 below presents the Agrovoc concept schema, based on SKOS model, where this schema is a top concept of all the others interpretation of Agrovoc model:

[92] In fact, AGROVOC is available as an SKOS-XL concept scheme.
[93] https://www.w3.org/RDF/
[94] https://www.w3.org/2004/02/skos/
[95] https://www.w3.org/wiki/LinkedData
[96] https://www.w3.org/TR/url/

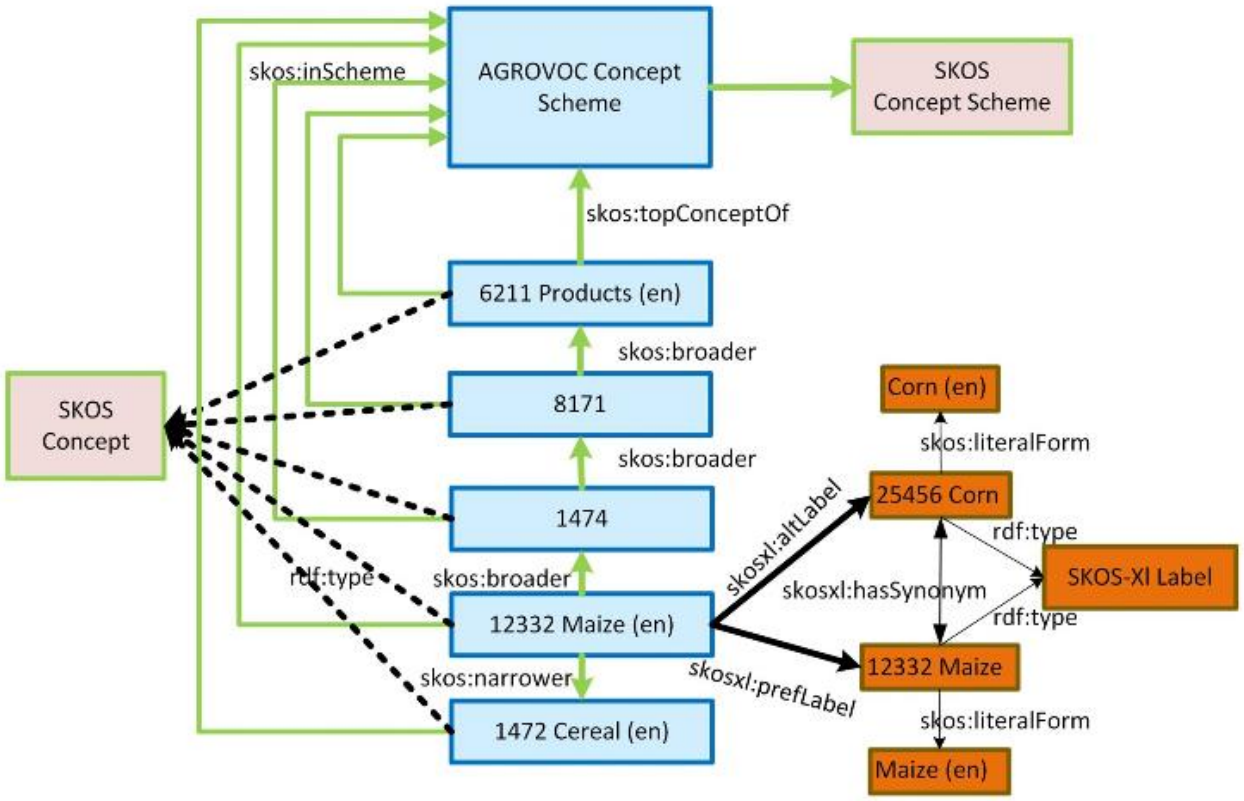


Figure 43. AGROVOC Concept Scheme [BoYa15]

With the introduction of Linking Open Data[97], W3C project for the extension of the Web with various open datasets such as RDF (using the concept of RDF link of data from heterogeneous resources), the Agrovoc ontology has also been integrated with the others Linked Data vocabularies, effectively enabling semantic interoperability with other datasets, other ontologies and so on. This is to remedy the related semantic problems that the Agrifood sector, like others, is called to face:

- heterogeneity of the actors along the supply chain using different standards and vocabularies;
- heterogeneity of data published by the various actors;
- integration of data within the supply chain with external data such as meteorological services and so on.

Using this approach, every actor in the chain, from producer to consumer, can publish his data and link it to other entities of other vocabularies and ontologies. Consequently, Agrovoc provides these semantically explicit structured data according to a vocabulary/ontology reference easily readable by a machine as a sensor and by a variety of smart devices that are increasingly used throughout the Agrifood supply chain. Semantic technologies

[97] https://www.w3.org/egov/wiki/Linked_Open_Data

then allow the integration of data into information systems, being specifically designed for this purpose.

Considering the large amount of data that DEMETER has to manage, it becomes increasingly difficult and expensive to query and analyze the information without creating a relationship between terms or entities. For instance, one can argue what are the internal elements of a system (components), by how this system is composed (e.g. devices), but if the relationships between the entities that represent the components, the devices are not modelled in the system, it is difficult to know which elements belong to each system of that type. There is no doubt that these entities must be modelled, like their relationships, but it is also necessary to establish a link between them and their semantic meaning by relating them to knowledge bases that perform this task.

This section deals precisely with this topic, that is to give an explanation and find a solution to the problem of semantic interoperability between the DEMETER AIM model (based on NGSI-LD) and the ontology model defined for Agrovoc. The primary objective in this case is to implement an alignment between AIM by creating relationships, or better of the references between the entity defined in the AIM ontology and the relative references in the vocabulary: this allow the entity to connect to its semantic meaning. The link takes place using the Linked Data: both NGSI-LD (and thus AIM) as well as Agrovoc provide interfaces of this type. All entities URI[98] and properties to which specific agricultural concepts refer (e.g. temperature, humidity, pressure, etc.) can be aligned by means of the property derived from the RDF Scheme *<rdfs:isDefinedBy>*[99] with the corresponding URI of Agrovoc (e.g. the one relating to temperature http://aims.fao.org/aos/agrovoc/c_7657).

NGSI-LD can support this mapping as its meta-model can connect one relationship to another through the Linked Data mechanism. This ontology, as already described extensively in the previous sections, is composed of two parts: a meta-model and an ontology between domains. The meta-model includes the following concepts: *Entity*, *Relationship*, *Property* and *Value*. The representation of the NGSI-LD meta-model reflects and extends the model *Entity-Relationship* (e.g. each relationship can be linked to another relationship, relationships can have properties) taking the form of property graph. There are many advantages in the use of NGSI-LD in relation to semantic interoperability (in this case with Agrovoc, but also with other types of vocabularies and ontologies):

- The data model is based on graphs and focuses on information. Information or entities can have properties and relationships. Instances of each entity can be the object of properties or relationships;
- Entities in NGSI-LD and their types can contain and be associated with a unique and unified URI (Uniform Resource Identifier) corresponding to semantic identifiers;
- Allow you to relate references to other vocabularies: all terms are defined unequivocally. This allows you to refer to information definitions;
- Use the JSON-LD data format which allows interconnection with connected data and therefore enables the unification of vocabularies;

[98] https://www.w3.org/wiki/URI
[99] https://www.w3.org/2000/01/rdf-schema#

- Allow the use of the *@context* syntax to include other ontologies and therefore the ontological references and the relationships between the entities of the model.

8.5.1 Data Elements Mapping Table

A comprehensive sample of relevant AGROVOC Object Model classes and potential mappings to DEMETER AIM classes are shown in Table 10.

Table 10. Mapping of AGROVOC OM Classes to AIM

AGROVOC Object Model Class	AIM Module	AIM Class
Animals	farmAnimal	FIWARE Animal Saref4agri s4agri:Animal, s4agri:AnimalGroup
Crop	AgriCrop	FOODIE:cropSpecies,cropType FIWARE:AgriCrop,Agrifood Saref4agri: Crop
Pests	AgriPest	FIWARE:AgriPest, FOODIE:Pest
Soil	agriProperty	FIWARE:AgriParcelRecord

Table 10 also lists the relevant AGROVOC Object Model classes that are suitable regarding the previously defined Domain-Specific ontologies in Section 7.3 using them AIM Classes could be extended with attribute which will refer to AGROVOC concept providing more specific information about entity.

8.6 Semantic Mapping to Earth Observation standards

General Requirements

The requirements for semantic interoperability for Earth Observation data arise from requirement "DK1.4 Earth Observation Data Representation" and the associated comment: "*OGC provides a suite of inter-related standards for EO data encoding and provision via services, however the semantic description aspects will require additional design work. This needs to be done in the context of a standards oriented meta-model that informs the DEMETER implementations so that consistency of approach can be achieved both within DEMETER and across other domains. OGC EO models, W3C Semantic Sensor Network and RDF-Datacube and other building blocks for this meta-model need to be adopted, adapted or mapped to in order to maximise the long-term value of DEMETER and allow re-use of software*".

Interoperability considerations

From the above statement and based on the pilots' analysis described in D5.1, a number of specific considerations can be identified and used to define a more detailed interoperability design:

1. The OGC is recognised as the relevant SDO defining interoperability specifications in this specific domain.
2. OGC specifications cover service interfaces, data structure, data encoding and spatio-temporal models of sensor deployment.
3. The OGC does **not** publish specifications that describe the subject matter of such observations, including what sensors are used, what is being sensed, the sample strategy, data processing etc. Consequently, DEMETER will need to identify and adopt or describe data products. In many cases, uses of the COPERNICUS and other data catalogs will be relevant.
4. Semantic descriptions need to cover both the documentation of data used, but also the data generated by DEMETER. Pilots may generate data products that use similar spatio-temporal distribution models to EO data, (gridded model outputs) and also products such as timeseries.

n-dimensional data semantics

5. OGC also publishes, jointly with the W3C, interoperability specifications related to Time – so to be consistent OGC should be regarded as the relevant SDO for temporal semantics, thus making spatio-temporal considerations easily integrated. These can be considered as **n-dimensional** data models.
6. OGC and other publish a range of encoding options for n-dimensional data products, including:
    a. coverageJSON: OGC 16-145: https://www.w3.org/TR/covjson-overview/
    b. https://binary-array-ld.github.io/netcdf-ld/
        - - in progress and LD-compatible, this is the recommended focus for data transfer
    c. Publishing and Using Earth Observation Data with the RDF Data Cube and the Discrete Global Grid System [https://www.w3.org/TR/2017/NOTE-eo-qb-20170928/]
        - An academic exercise, but shows how SOSA/RDF-QB and QB4ST can be used for small amounts of EO data
7. A discussion paper on options for JSON encodings for coverages is at http://docs.opengeospatial.org/dp/19-042r1.html
8. A discussion paper on use of OGC models in JSON-LD is at https://github.com/opengeospatial/architecture-dwg/blob/eefba7e2f250fd82b47ae4dc546525eb89eb1ecc/json-best-practice/clause_7-from_json_to_jsonld.adoc#json-for-coverages
9. Current activity inside the OGC community on consolidating the issues raised in these discussions in a Linked Data context is the "Binary-Array-Ld" activity [https://binary-array-ld.net/]. This is recommended as the initial focus for DEMETER for EO and gridded model outputs, as it integrates naturally with the JSON-LD basis of NGSI-LD and DEMETER.

Interoperability with observable phenomena descriptions in DEMETER data model

10. OGC does not publish semantic resources for domain specific phenomena, such as observable properties. The terms and definitions from the original EO data providers will be used, and if necessary DEMETER will publish these as web accessible semantic resources.
11. It will be necessary in general to profile OGC specifications to bind them to the choice of DEMETER defined (or adopted) phenomena descriptions in order to create semantic models of EO and other n-dimensional data components in DEMETER pilots.
12. RDF-Datacube is inherently suited to describing n-D data and can be profiled to provide an interoperable way of binding observation semantics to both n-D data and collections of in-situ sensor and model outputs using the DEMETER AIM.
13. OGC published a draft profile of RDF-Datacube (QB4ST https://www.w3.org/TR/qb4st/ - document number OGC 16-142) – this is a W3C Note and the rechartering of Spatial Data on the Web Working Group means that this can be readily updated to meet DEMETER needs, particularly since the author is a DEMETER participant.

Catalogue related requirements

14. For EO data Catalogs DK 1.4 has specified use of the specification: "OGC 13-026r8: OGC OpenSearch Extension for Earth Observation". This is a reasonable starting point, however, may need to be extended to handle related forms of data derived from EO and other data processing processes.
15. Semantic descriptions of data are proscribed for cataloguing purposes and include Tables 3,5 & 6 which define EO domain specific metadata elements, and Table 4 which describes the set of parameters required to meet INSPIRE requirements. Thus, this specification may be considered to be a profile of an INSPIRE profile of OpenSearch. This is appropriate for DEMETER context.
16. The parameters in this specification need to be mapped to equivalent elements in the DEMETER data model. Thus, it is recommended that the DEMETER data model uses the set of definitions as the basis for EO data models, and that OGC is tasked to publish canonical URI identifiers for each of these terms.
17. A key parameter defined sru:recordSchema which defines the "metadata model" which should be provided in a response. Thus, there are three separate elements of interoperability required:
    a. service API (OpenSearch)
    b. query model (OGC 13-026r8 supported query profiles)
    c. response payload model (e.g., http://www.opengis.net/sensorml/2.0)
    d. response payload encoding
18. To be consistent with DEMETER architecture, JSON-LD should be supported as the preferred ("canonical") encoding option where available from catalogue providers, however individual tools may choose other supported models and encodings.
19. DEMETER should identify the preferred payload schemas required by pilots and define (with an identifying URI) a "DEMETER data catalogue profile" of 03-026r8) to make this design choice explicit.

20. DEMETER should use an external catalog or deploy one to describe all DEMETER generated data products, conforming to the “DEMETER data catalogue profile”
21. The DCAT vocabulary provides a canonical means to integrate n-D data semantics into cataloguing approaches, consistent with the CYBELE data model.

Given these considerations, DEMETER needs to enable the representation of current earth observation (EO) data as well as historical EO data. This will be aligned with existing practice around the use of OGC WCS and GeoJson payloads by creation of a “modernised” approach to WCS incorporating NGSI-LD compatatible JSON-LD encoding and the SOSA model for observation metadata via feature models based on the DEMETER AIM, as per the diagram in the figure below, which highlights the use of the AIM meta-model in implementing EO data descriptions. Note this diagram does not show it directly, but also reflects the application of the same approach to the use of DCAT for descriptions of EO data, where NGSI-LD would be replaced with the preferred catalog service APIs.

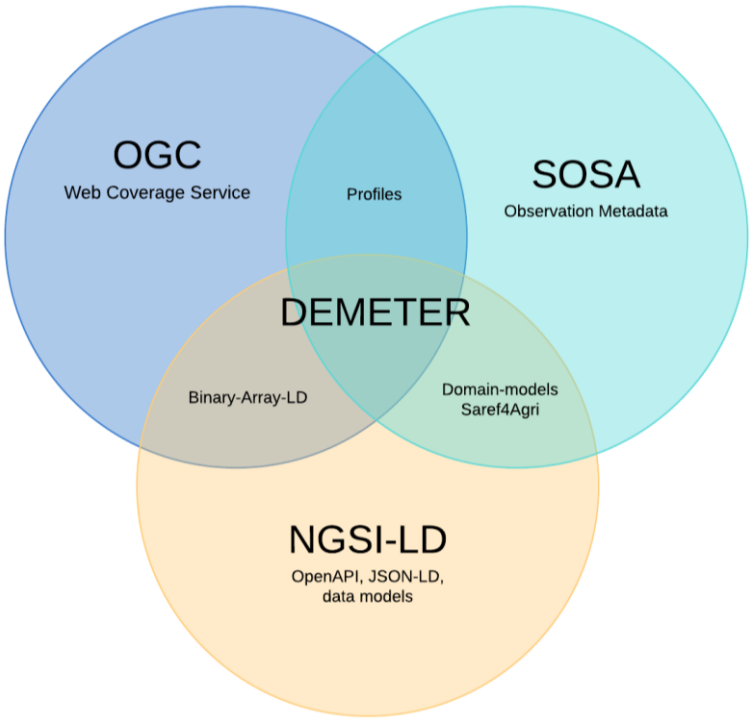


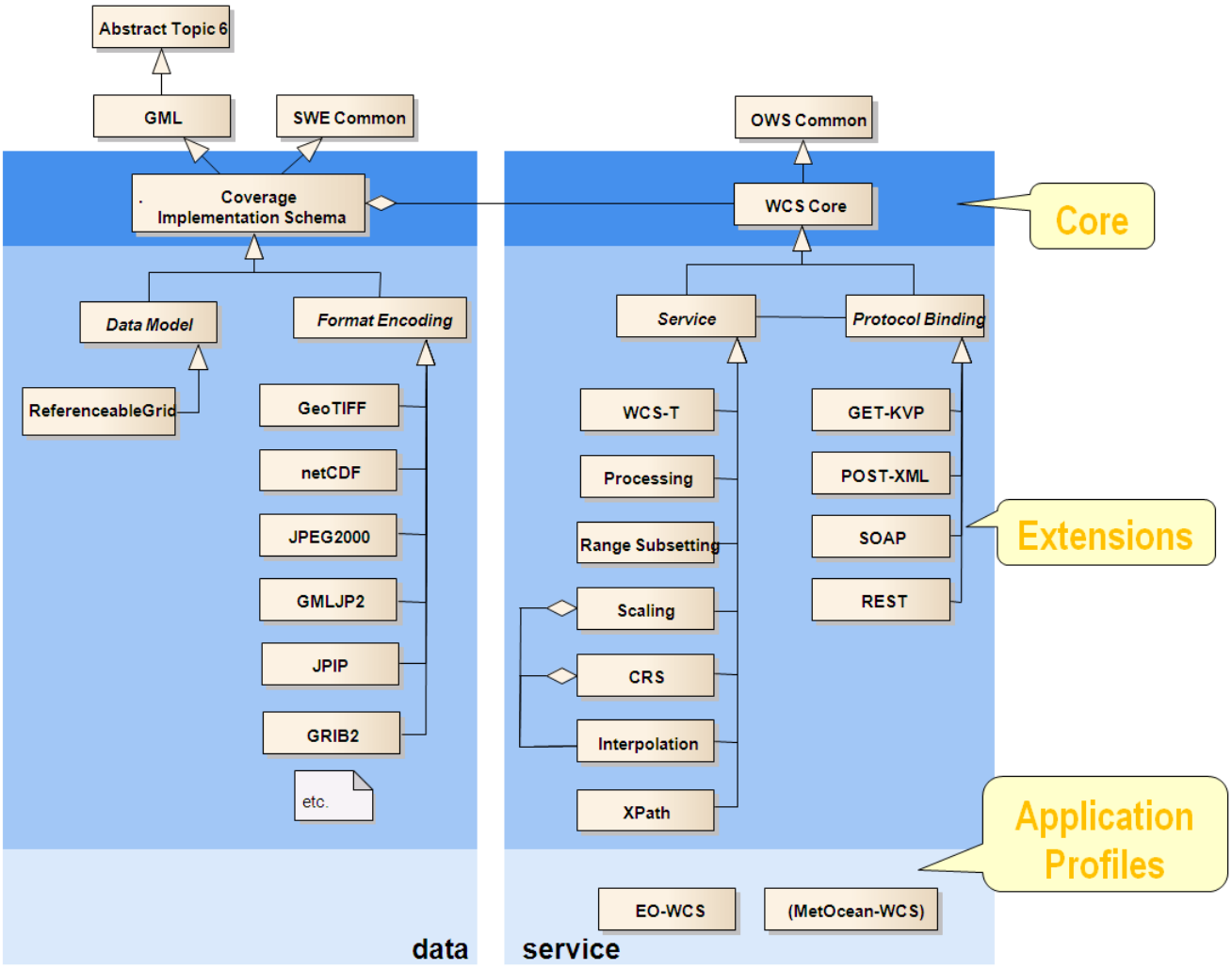
Figure 44. Semantic interoperability for Earth Observation data in DEMETER

Some components of DEMETER pilots for EO already follow the OGC standard using GeoJSON, and even include a @context for GeoJSON.

The DEMETER project will improve the interoperability of EO data by introducing best practices to combine domain specific models (AIM) with available data exchange implementations used for EO data.

The OGC already separates the concerns between an “Abstract Model” and implementations, such as WCS.


```
{
      "@context":"http://schemas.opengis.net/os-geojson/1.0/os-geojson.jsonld",
      "type'": "FeatureCollection",
      "id":
"https://services.terrascope.be/catalogue/products?collection=urn%3Aeop%3AVITO%3ATE
RRASCOPE_S2_FAPAR_V2",
      "features":
      {
            "type": "Feature",
            "id":
"urn:eop:VITO:TERRASCOPE_S2_FAPAR_V2:S2B_20191227T105349_31UFS_FAPAR_10M_V200",
            "geometry":
            {
                  "type": "Polygon",
                  "coordinates":
            }
      },
      " properties":
      {
            "date": "2019-12-27T10:53:49Z",
            "updated": "2020-04-09T20:21:06Z",
            "available": "2020-04-16T15:00:32Z",
            "published": "2020-04-16T15:00:32Z",
            "status": "ARCHIVED",
            "parentIdentifier": "urn:eop:VITO:TERRASCOPE_S2_FAPAR_V2",
            "title": "S2B_20191227T105349_31UFS_FAPAR_10M_V200",
            "identifier":
"urn:eop:VITO:TERRASCOPE_S2_FAPAR_V2:S2B_20191227T105349_31UFS_FAPAR_10M_V200",
            "acquisitionInformation":
            {
                  "platform":
                  {
                        "platformShortName": "SENTINEL-2",
                        "platformSerialIdentifier": "S2B",
                  },
            },
      },
},
```



Following this pattern, DEMETER defines the "data model" using the AIM model, and define and adopt a profile of the OGC WCS that uses the NGSI-LD compatible encoding of the AIM model, with its basis in the OGC/W3C sosa:Observation.

If the opportunity presents, this profile should be aligned with the planned "next generation" OGC API – Coverages specification [https://github.com/opengeospatial/ogc_api_coverages] to simplify alignment with NGSI-LD by using the OpenAPI model instead of the legacy WCS standard. This provides the opportunity for a significant impact on interoperability standards based on DEMETER pilot experience.

9 Implementation of AIM and of Semantic Mappings

DEMETER AIM has been implemented following a layered and modular approach, reusing as much as possible existing ontologies and vocabularies, as described in the previous sections. In this section, we introduce the implementations for the different layers (parts) of AIM, along with a discussion of the design and implementation choices taken, the mappings implemented and the tools used during the implementation process. Links to the final results (i.e. the files implementing the AIM) are also provided.

9.1 General considerations

In general, DEMETER AIM modules have been implemented as OWL ontologies, and serialized as Turtle.[100] Furthermore, we transformed the modules into JSON-LD contexts. JSON-LD enables the encoding of linked data in JSON, one of the most commonly used formats to exchange data between services; it also helps JSON data to interoperate at Web-scale. The context in JSON-LD is used to map terms, i.e., properties with associated values in a JSON document, to URIs (Uniform Resource Identifiers), such as OWL entities. JSON-LD contexts allow disambiguating keys shared among different JSON documents by mapping them to URIs which describe their meaning: two applications can use shortcut terms to communicate together more efficiently, without losing accuracy.

For the implementation of the ontologies and related contexts, we used different tools. Some of the ontologies were handled manually using a simple text editor; we also used tools like Protégé[101] to facilitate the implementation or to visualise and navigate the ontologies. We validated the ontologies using the reasoners in Protégé[102]. In particular, we used the Pellet reasoner to verify the logical consistency of the ontologies. This was particularly useful to identify issues/inconsistencies across different modules. For example, the DEMETER AIM agri profile imports over 10 modules, and making the logical validation over it allowed us to identify and solve some inconsistencies that we could not spot manually.

In order to transform the ontologies generated into JSON-LD contexts (to enables services to exchange JSON data about the different entities), we used the tool owl2jsonld[103]. This tool generates a JSON-LD @context for concepts (classes and properties) found in the specified OWL or RDFS ontology. The script to generate the contexts for the Agri-modules is available at: https://raw.githubusercontent.com/rapw3k/DEMETER/master/models/jsonld/auto-generate-context.sh.

All the ontologies generated, as well as the corresponding JSON-LD contexts, use persistent identifiers that are resolvable. This facilitates both the sharing and usage within different applications, through time. We decided to

[100] https://en.wikipedia.org/wiki/Turtle_(syntax)
[101] https://protege.stanford.edu/
[102] https://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI
[103] https://github.com/stain/owl2jsonld

use w3id[104] service for permanent identifiers on the Web. The service, runned by the W3C permanent identifier community group, provides secure, permanent URL re-directions for Web applications. As a result, DEMETER resources will *always* be accessible and resolvable, even if the physical locations of the resources change. The base namespace for DEMETER AIM is: https://w3id.org/demeter/ (which resolves to DEMETER GitLab code repository at https://gitlab.com/demeterproject/).

Regarding mappings between different vocabularies/ontologies, DEMETER AIM defines them in each module by including appropriate ontology axioms, such as equivalent classes (owl:equivalentClass), equivalent properties (owl:equivalentProperty), subclasses (rdfs:subClassOf) and subproperties (rdfs:subPropertyOf). We mostly defined the mappings manually, but we also tried or used some tools for verification. The initial results are not very encouraging, but we plan to evaluate them further in a second stage. For example, the Alignment API server[105] , well-acknowledged from the ontology community, was initially tested in order to map the saref4Agri ontology and the FOODIE ontology. However, this exercise did not give many useful mappings, except for 3 interesting (2 of them were already in the manual mappings). This is because the names of the entities are too different between the ontologies, and the most common mapping methods are based on string similarity (FIWARE modules did not have any ontology implemented, so it had to be mapped manually). Nevertheless, this tool implements many matching methods, so it would be quite interesting experiment to test all possibilities in the future. In the simplest case, the tool finds exact string matches (using name/label of entities). A more advanced way is to tell the tool to find not only exact matches but also to use string similarity methods (e.g., levenshteinDistance), and to specify the threshold of confidence of the matches (in our case lower than 0.6 gave totally useless results). Other methods are listed under https://github.com/dozed/align-api-project/tree/master/procalign/src/main/java/fr/inrialpes/exmo/align/impl/method . The command used for the mapping and the results obtained are illustrated using in the code presented below.

java -jar lib/procalign.jar -i fr.inrialpes.exmo.align.impl.method.StringDistAlignment -DstringFunction=levenshteinDistance -t 0.6 file:///Users/rap/Downloads/align-4.9/ontos/foodie.ttl file:///Users/rap/Downloads/align-4.9/ontos/ontology.ttl

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<rdf:RDF xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment#'
         xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema#'
         xmlns:alext='http://exmo.inrialpes.fr/align/ext/1.0/'

xmlns:align='http://knowledgeweb.semanticweb.org/heterogeneity/alignment#'>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
```

[104] https://w3id.org/
[105] http://alignapi.gforge.inria.fr/

```
  <type>?*</type>

<alext:method>fr.inrialpes.exmo.align.impl.method.StringDistAlignment</alext:method
>
  <alext:time>90</alext:time>
  <onto1>
    <Ontology rdf:about="http://foodie-cloud.com/model/foodie">
      <location>file:///Users/rap/Downloads/align-4.9/ontos/foodie.ttl</location>
      <formalism>
        <Formalism align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#"/>
      </formalism>
    </Ontology>
  </onto1>
  <onto2>
    <Ontology rdf:about="https://w3id.org/def/saref4agri">
      <location>file:///Users/rap/Downloads/align-4.9/ontos/ontology.ttl</location>
      <formalism>
        <Formalism align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#"/>
      </formalism>
    </Ontology>
  </onto2>
  <map>
    <Cell>
      <entity1 rdf:resource='http://foodie-
cloud.com/model/foodie/code/PropertyTypeValue/soilTexture'/>
      <entity2 rdf:resource='https://w3id.org/def/saref4agri#SoilMoisture'/>
      <relation>=</relation>
      <measure
rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.6666666666666667</measure>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://foodie-cloud.com/model/foodie#containsZone'/>
      <entity2 rdf:resource='https://w3id.org/def/saref4agri#contains'/>
      <relation>=</relation>
      <measure
rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.6666666666666667</measure>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://foodie-cloud.com/model/foodie#containsPlot'/>
      <entity2 rdf:resource='https://w3id.org/def/saref4agri#contains'/>
      <relation>=</relation>
```

```
      <measure
rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.6666666666666667</measure>
    </Cell>
  </map>
</Alignment>
</rdf:RDF>
```

9.2 DEMETER AIM meta-model

DEMETER AIM adopts and reuses NGSI-LD meta-model, which provides a formal basis for representing "property graphs" using RDF/RDFS/OWL. It allows back and forth conversion between datasets that are based on the property graph model and linked data datasets, which rely on the RDF framework.

In particular, the meta-model defines the following entities (adapted from NGSI- LD [ETS18]):

- **Entity**: A DEMETER entity is defined as an NGSI-LD Entity, which is the informational representative of something that is supposed to exist in the real world, physically or conceptually. Any instance of such an entity shall be uniquely identified by a URI, and characterized by reference to one or more NGSI-LD Entity Type(s).
- **Property**: A DEMETER property is defined as an NGSI-LD property, which is a description instance that associates a main characteristic, which shall be a DEMETER Value, to either a DEMETER Entity, a DEMETERRelationship or another DEMETER Property. It shall include the special "hasValue" property to define its target value.
- **Value**: A DEMETER value is defined as an NGSI-LD Value that is either a JSON value (i.e. a string, a number, true or false, an object, an array), or a JSON-LD typed value (i.e. a string as the lexical form of the value together with a type, defined by an XSD base type or more generally a URI), or a JSON-LD structured value (i.e. a set, a list, a language-tagged string).
- **Relationship**: A DEMETER relationship is defined as an NGSI-LD Relationship that describes a directed link between a subject, which shall be either a DEMETER Entity, a DEMETER Property, or another DEMETER Relationship on one hand, and an object, which shall be a DEMETER Entity, on the other hand. It shall include the special "hasObject" property to define its target object.

The meta-model has been implemented as a JSON-LD context in the same way as the NGSI-LD one (enabling the encoding of linked data in JSON). However, instead of defining cross-domain terms in the same context, as NGSI-LD does, the DEMETER meta-model is limited to the entities enabling the representation of “property graphs” described above.

DEMETER AIM meta-model implementation is available at: https://w3id.org/demeter/core-context.jsonld

9.3 DEMETER AIM cross-domain ontology

The DEMETER AIM Cross-Domain Ontology is a generic model and aims to build a bridge between the different ontologies present in DEMETER. It reuses standardized and widely-used vocabularies in order to link between ontologies horizontally as well as between different layers of there vertically. Sections 7.1 and 7.2 present the concept and the resulting implementation is available online[106] in Turtle format (*.ttl), along with a standardized documentation and visualization.

As stated in section 7.2, we reuse existing ontologies such as SOSA and OWL to achieve cross-domain linking via axioms. Where things are exactly the same in the DEMETER AIM context, we use owl:equivalentProperty and owl:equivalentClass for properties and classes, respectively. See the example below for the properties stapi:description and dct:description, as well as stapi:Datastream and oboe:ObservationCollection.

```
###  http://www.opengis.net/doc/is/sensorthings/1.0/description
stapi:description rdf:type owl:ObjectProperty .

###  http://purl.org/dc/terms/description
dct:description rdf:type owl:ObjectProperty ;
    owl:equivalentProperty stapi:description .

###  http://www.opengis.net/doc/is/sensorthings/1.0/Datastream
stapi:Datastream rdf:type owl:Class .

###  http://ecoinformatics.org/oboe/oboe.1.0/oboe-core.owl#ObservationCollection
oboe:ObservationCollection rdf:type owl:Class ;
    owl:equivalentClass stapi:Datastream .
```

Sometimes a mapping between classes is not suitable, but rather between its instances – called “individuals” in OWL. We therefore integrate the owl:sameAs property to this cross-domain ontology, see the example below where the same sensor data was stored twice and is connected again via this statement.

```
###  https://w3id.org/demeter/cross-domain#Observation1Backup
:Observation1Backup rdf:type owl:NamedIndividual .

###  https://w3id.org/demeter/cross-domain#Observation1
:Observation1 rdf:type owl:NamedIndividual ;
              owl:sameAs :Observation1Backup .
```

Following other standards, we enable vertical linking within one ontology, and also between layers of different ontologies. These functionalities shall be used to denote more general concepts (rdfs:subClassOf) as well as more general relationships (rdfs:subPropertyOf). The inverses are handled automatically in applications by traversing

[106] https://w3id.org/demeter/cross-domain

the properties backwards. An example for vertical linking is shown in the following example, which (following section 5.5) states that the saref:Device is a specific ssn:System. Another example is the alignment that the windSpeed property in Fiware Weather is a particular hasValue property from saref.

```
ssn:System rdf:type owl:Class .

saref:Device rdf:type owl:Class ;
    rdfs:subClassOf ssn:System .

saref:hasValue rdf:type owl:DatatypeProperty .

###  https://github.com/smart-data-models/dataModel.Weather/windSpeed
fiweather:windSpeed rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf saref:hasValue .
```

We use dcterms:source for a related resource from which a resource is derived. The following example links the concept ProductType in ADAPT to the Atlassian web page[107] where it was discussed and defined. If we need to reference a complex resource, we use prov:wasDerivedFrom instead. In the same example, we present the usage of skos:notation for uniquely identifying a concept within the scope of a given scheme.

```
###  http://w3id.org/demeter/ADAPT-mapping/#ProductType
adapt:ProductType rdf:type owl:NamedIndividual ;
    dct:source <https://aggateway.atlassian.net/wiki/spaces/[...]Discussion> ;
    skos:notation "1001" .
```

As a more precise alternative to dcterms:source as mentioned above, we can also follow the example of AgriProperty[108] and use rdfs:isDefinedBy. The difference to dcterms:source is that the object of dcterms:source can even be a string (e.g., a literature reference), whereas the object of rdfs:isDefinedBy has to be a URI.In cases where we need our own documentation or explanation of used terms, we should use, like in the other implementation modules, rdfs:label and rdfs:comment. In case things have multiple labels, we can use skos:prefLabel along with multiple skos:altLabel.

If the respective notion of "concept" is not logically as strict as that of an OWL/RDFS class (see the first example in this section), of which instances exist, but rather like a subject/field/theme, with which things may be associated (like books in a library classification scheme), then we can instead use SKOS as follows. Properties or classes of existing ontologies to be reused (SOSA, QUDT etc.) might not currently be declared as instances of

[107] https://aggateway.atlassian.net/wiki/spaces/ADM/pages/57672166/ADAPT+-+Product+CropProtectionProduct+FertilizerProduct+CropVariety+Trait+Discussion
[108] https://raw.githubusercontent.com/rapw3k/DEMETER/master/models/agriProperty.ttl

skos:Concept. The following code that makes them instances of skos:Concept via a mapping[109], which however is not harmful. The benefits are stronger than this "drawback".

```
aim:ourConcept skos:exactMatch existing:Concept ;
dcterms:source … # as above
```

We use skos:closeMatch instead, if the match is not exact but close. Additionally, we use skos:broadMatch if the existing concept is broader (i.e., more general) than ours. We use skos:narrowMatch if the existing concept is narrower (more specific) than ours. We use skos:relatedMatch if they are somehow related but it's not clear how exactly. For exemplary usage, please refer to the previous example and replace skow:exactMatch with the respective relation.

The DEMETER AIM cross-domain ontology is a collection of important standards and de-facto standards (cf. section 7.2) that should be seen as "tools" to tie together the different ontologies in DEMETER, both horizontally and vertically. The current state models the above-mentioned examples and is intended to grow incrementally (e.g., via commits in the Git repository) in order to improve the cross-domain knowledge in DEMETER over time.

9.4 DEMETER AIM domain-specific ontologies

DEMETER AIM covers different domains of the agrifood sector identified in the requirements sections, which are relevant for the development and support of different smart farming-IoT based platforms and solutions. In the following, we describe the different domains under the agrifood sector as these are covered in the first version of the model.

DEMETER AIM implemented different modules to cover the entire agri-food sector concepts. These modules have been inspired by the FIWARE agri-food models, but also took into account the structure of the main ontologies identified. In particular, the main ontologies and models re-used for this domain include:

- Saref4Agri (which extends SAREF, and reuses SSN and SOSA ontologies among others)
- FOODIE (which extends INSPIRE agriculture and aquaculture facilities model, and reuses ISO standards)
- FIWARE agri-food models (which are aligned with the NGSI-LD model). It is important to note though, that these models are mainly available as documentation and/or json schemes, along with some examples of instances using them in JSON and JSON-LD. Furthermore, the terms defined in these (and other) models are also available in one large (mostly flat) FIWARE JSON-LD context[110],, which maps terms names to URIs. However, these URIs are not further defined explicitly in an ontology specifying meaning and semantics in machine-readable format.

[109] https://www.w3.org/TR/skos-reference/#mapping
[110] https://fiware.github.io/data-models/context.jsonld

The general approach was to use SAREF4Agri as main source given its good documentation, structure and coverage, and extend with FOODIE and FIWARE entities. In order to reuse the entities from these ontologies, we followed the best practices in ontology engineering for reusing ontology statements (http://neon-project.org/web-content/media/book-chapters/Chapter-11.pdf). Reusing ontology statements instead of whole ontologies may be useful in different cases as the reuse of large ontologies can be difficult because they contain a large amount of knowledge that may not be needed when developing a particular ontology. Also, sometimes, reuse demands to retrieve pieces of knowledge (e.g., statements) to be integrated in the new ontology being built rather than to reuse entire ontologies. In our case, given our modular approach, reusing ontology statements was better option to keep modules self-contained and focussed, while at the same time facilitating their maintenance and the extensibility of the DEMETER AIM.In order to reuse statements, we copied (most of) the statements from the original ontologies into the DEMETER module, and we included the link to the ontology where the entity is defined using the property: rdfs:isDefinedBy. This allows getting full definition of the entity, tracing it back to its source, while at the same time allows an easy integration in the module. Note that in the case of FIWARE entities, we had to define entities from scratch in our modules, as these were not available in any ontology (as discussed above).

After the reuse (or creation in the case of FIWARE) of the different relevant terms, mappings were added in the module to align the three ontologies/models, as described in Section 9.1.

In particular, DEMETER AIM agri-food profile is available at https://w3id.org/demeter/agri and imports the following modules:

- agriCommon.ttl[111]: module that includes common properties used across all other agri-food modules.
- agriProperty.ttl[112]: module focused on the different agri properties measured/observed in agri-food applications (e.g., temperature, humidity, etc.) and their connection to the systems used to collect them.
- agriSystem.ttl[113]: module including all the entities to represent and describe systems and platforms related to agri-food sector, e.g. irrigation system, weather stations, etc., including particular sensors in these systems.
- agriAlert. ttl[114]: module enabling the representation of agri-food alerts, and their characteristics.
- agriCrop. ttl[115]: module focused on the representation of crops and their characteristics.
- agriFeature.ttl[116]: module enabling the representation of geo features relevant to agri-food applications, e.g., farms, plots, etc.

[111] https://w3id.org/demeter/agri/agriCommon
[112] https://w3id.org/demeter/agri/agriProperty
[113] https://w3id.org/demeter/agri/agriSystem
[114] https://w3id.org/demeter/agri/agriAlert
[115] https://w3id.org/demeter/agri/agriCrop
[116] https://w3id.org/demeter/agri/agriFeature

- agriIntervention.ttl[117]: module including entities to represent and describe different agri interventions, e.g., fertilization, irrigation, etc.
- agriPest.ttl[118]: module enabling the representation of agri pests, and their characteristics.
- agriProduct.ttl[119]: module focused on the representation of agri-food products and their characteristics, e.,g., nutrients, ingredients, etc.
- farmAnimal.ttl[120]: module focused on the representation of data about livestock, and their characteristics.

9.5 DEMETER AIM metadata schema

For the metadata schema implementation, a fork of the IDS Information Model (refer to section 5.15.3) is created and adapted to suit the particular needs of DEMETER as described in section 7.4. For this, the IDS Informationen model if forked and copied fromfrom IDS GitHub: https://github.com/International-Data-Spaces-Association/InformationModel/ to DEMETER GitLab: https://gitlab.com/demeterproject/wp2/agriculturalinformationmodel/metadataschema

More detailed documentation on the implementation can be found in the respective DEMETER GitLab respository. The DEMETER fork of the IDS InformationModel will be maintained by DEMETER and in order to benefit from improvements and new features of the original IDS Information Model, which is actively mainained, a merge from the original Repository into the DEMETER metadata repository will be performed on a regular basis. Here it should be ensured that new IDS features do not break with DEMETER requirements and/or the DEMETER fork will be adapted accordingy.

The implementation and documentation of the original IDS infomodel is available at: https://w3id.org/idsa/core

The resulting DEMETER medata schema will be available at: https://w3id.org/demeter/metadata

And an example instance of the DEMETER medata schema is available at: https://gitlab.com/demeterproject/wp2/agriculturalinformationmodel/metadataschema/-/raw/master/metadata-instance.ttl

[117] https://w3id.org/demeter/agri/agriIntervention
[118] https://w3id.org/demeter/agri/agriPest
[119] https://w3id.org/demeter/agri/agriProduct
[120] https://w3id.org/demeter/agri/farmAnimal

10 Conclusions

This deliverable describes in detail the initial release of the DEMETER Common Data Models and Semantic Interoperability Mechanisms and presents the DEMETER Agricultural Information Model (AIM) (Release 1), which will be the data model to be used by all the first round DEMETER pilots. It initially presents an analysis of the State of the Art on related data models and interoperability mechanisms that are suitable for the domain of smart agrifood. Subsequently, it presents the technical requirements extracted by Task 2.1, which drive the design and development of DEMETER AIM and semantic interoperability mechanisms. Followingly, it presents in detail the initial release of the DEMETER Agricultural Information Model (AIM) design, which adopts a modular approach and is based on a rich set of related standards or dominant solutions. AIM distinguishes between four main parts, each with a different role: the ***AIM core metamodel***, which builds on and extends the NGSI-LD meta-modelling approach; the ***AIM cross-domain ontology***, i.e., the set of generic models that aim at providing common definitions for not necessarily tied to the agrifood sector and at avoiding conflicting or redundant definitions of the same classes at the domain-specific layer; the ***AIM domain-specific ontologies*** that model information related to all domains linked to the agrifood section, such as crops, animals, agricultural products, as well as farms and farmers just to mention a few of the most important concepts included in these ontologies; and finally, the ***AIM metadata schema*** that aims to represent and capture any metadata that may be required by DEMETER. Next, it elaborated on the interoperability support provided by the DEMETER AIM with regards to several existing standards and dominant agri-food data modelling approaches (such as NGSI-LD and FIWARE, Saref4Agri, ADAPT, INSPIRE and FOODIE, AGROVOC and EO data) detailing the semantic mapping of these to AIM. Subsequently, it presents the development process and tools used for the implementation of the first release of AIM, providing the references to the respective software developed. Finally, the document concludes with a summary, discusses the ongoing activities related to AIM that aim to support the first round of the DEMETER pilots and the future plans that are in place in view of the final release of AIM. In this respect, Annex A records all data types that will be engaged in the 20 DEMETER pilots, as these have been reported by WP5.

The content of this deliverable is the result of collaborative work of partners not only in Task 2.1 (that is responsible for D21), but also of the other tasks in WP2, as well as some input from WPs 3, 4 and 5. More specifically, the technical requirements (Section 6) took into account input from these WPs and especially WP5, while the annex is based on input provided by the pilot leaders in WP5.

This deliverable contributes to the achievement of Milestone 2 (DEMETER Enablers, Hub, Spaces and Applications Release 1) planned for June 2020.

Herafter, we present some ongoing and future work that is scheduled to be completed for the final release of the AIM (that will be presented in D2.3 next year). This work can be split into two broad categories: the further development and evolution of AIM and extending its interoperability support to address additional existing data modelling approaches.

One of the elements not fully addressed in this release of AIM, is the full range linking of the four AIM parts (i.e.,

the core metamodel, the cross-domain ontology, the domain-specific ontologies and the metadata schema). The establishment of the respective missing connections among these four are of high priority for WP2 and currently lies among the objectives of ongoing work, aiming for these connections to be in place on time for the first round of pilots. Of course, all four parts will continuously evolve and extended versions of the ones available at this point will be in place to support the initial pilots.

Regarding the extension of AIM, currently, the information related to agricultural machinery in the domain-specific ontologies is limited to a few classes reused from FOODIE in the *agriSystem* ontology. However, in the agrifood sector, more types of vehicles/machinery are used, including UGVs and UAVs, which requires modelling a vehicle hierarchy. Furthermore, currently, we have information such as engine data, fuel consumption, emissions, etc., which can be modelled as observations using SSN/SOSA; but it is also necessary to model information related to the tasks that the vehicle/machinery can carry out (e.g., commands related to the implements of an ISOBUS tractor, trajectories of a UAV to scan an area, follow a target or detect and object). In fact, ISOBUS is a very well known and used standard and we plan to use concepts from it in the extension of the AIM machinery model. Another potential source for this extension that we aim to investigate is CEMA's (European Agricultural Machinery) recent initiative for the deployment of agricultural machinery data-sharing.[121]

There are also a few requirements identified in Section 6.1 which are not addressed in the current version the domain-specific ontologies of AIM. The first is linked to traceability, for which we currently investigate reusing terms and entities from the FOODON ontology in order to enrich AIM with support for traceability, as well as using location from the WGS84 Geo Positioning ontology, in order to develop a (smaller than FOODON) ontology that covers this need. In this respect, an additional domain-specific ontology is foresen, about which the GS1 EPCIS is also being investigated to be exploited by the DEMETER AIM in support of agrifood product traceability. Second, we also need to enable representation of farmers' characteristics, policies, needs and preferences. So far, the current release of AIM has only limited information about the farmers and details about their preferred farm operations configurations and priorities. To address this, a new domain-specific ontology will be developed; however, as there is no ideal data model DEMETER can build on, we will use the pilots to examine the farmers' needs in more detail, thus determining what data this ontology needs to cover. Third, there maybe the need to treat all inputs used in farm operations (e.g., energy, water, pesticides, fertilisers, etc.) differently than other resources handled by the system. These are currently captured under the agriProperty or the agriProduct ontologies, but it is under inverstigation whether an additional ontology is required for these inputs, enriching these with extra features and metrics, useful for benchmarking and decision making support, e.g., financial parameters, resource consumption, etc.

Furthermore, Earth Observation concepts and derived data are not extensively represented in the current release of AIM, i.e., as mainly only related data from SSN/SOSA are now part of the AIM design. However, should

[121] http://www.innoseta.eu/wp-content/uploads/2020/02/2020_02_05_CEMA-PT3-PP_Strategy-paper-agricultural-machinery-data-sharing.pdf

an AIM full scale EO model becomes necessary, we plan to extend the cross-domain ontology of AIM using OGC's Coverage abstract model, coupling this with the SOSA/SSN Observation model and an appropriate data record grid encoding. This will be compliant with the OGC 13-026r8 standard and will be used to promote improved relevant standards on EO data modelling.

Regarding the further extension of semantic interoperability support, AIM is to be further extended with additional terms and concepts defined by AGROVOC. For example, by adding the `hasAgriVocConcept` in domain-specific ontologies of AIM, such as its agriCrop ontology, links to AGROVOC will be strengthened, while the establishment of any further connections will also be investigated.

We will also investigate the potential linking of the crop types used in the AIM Crops ontology with EU initiatives outcomes that have recently delivered such crop type classifications (e.g., the H2020 NIVA project), as well as the UN standard "eCROP"[122] regarding the representation of crop types, other relevant data input collected during farm operations, etc.

Moreover, this work is to be complemented by the establishment of best practices and broader community around formalising metadata profiles to enhance data interoperability across heterogenous systems and for managing domain models in complex multi-stakeholder domains, particularly those spanning multiple standards frameworks.

Finally, we are working closely with Task 2.2 in order to implement enablers based on the current version of AIM that will support the actual translation of data from/to AIM format to/from the various dominant ontologies aforementions, e.g., NGSI-LD/FIWARE, Saref4Agri, FOODIE, ADAPT, AGROVOC, etc. The progress of this work will be reported in D2.2 and carry on during the pilot execution as well.

Finally, in a year from now, the revised version of the DEMETER Common Data Models and Semantic Interoperability Mechanisms is planned for release that will be presented in D2.3 to be delivered in April 2021.

[122] https://www.unece.org/fileadmin/DAM/cefact/brs/BRS_eCROP_v1.pdf

11 References

[AC13] European Commission, “INSPIRE Data SpecificationS – Base Models – Activity Complex” [online], 2013, available at URL <http://inspire.ec.europa.eu/documents/Data_Specifications/D2.10.3_Activity_Complex_v1.0rc3.pdf >

[AF13] European Commission, “Data Specification on Agricultural and Aquaculture Facilities – Technical Guidelines” [online], 2013, available at URL <https://inspire.ec.europa.eu/documents/Data_specifications/INSPIRE_DataSpecification_AF_v3.0.pdf>.

[Agr] "AgroXML," [Online]. Available: http://195.37.233.20/.

[BoYa15] Boulos, M., Yassine, A., Shirmohammadi, S., Namahoot, C., & Brückner, M. (2015). Towards an “Internet of Food”: Food Ontologies for the Internet of Things. Future Internet, 7(4), 372–392. doi:10.3390/fi7040372

[DCAT20] “Data Catalog Vocabulary (DCAT) - Version 2” 2020-02. [Online]. Available: https://www.w3.org/TR/vocab-dcat-2/.

[DQV16] “Data on the Web Best Practices: Data Quality Vocabulary” 2016-12. [Online]. Available: https://www.w3.org/TR/vocab-dqv/.

[EIF04] "European Interoperability Framework (EIF), White paper, pp. 1-40," 2004. [Online]. Available: http://www.urenio.org/e-innovation/stratinc/files/library/ict/15.ICT_standards.pdf.

[ETS18] "ETSI GR CIM002 V1.1.1 Context Information Management (CIM); Use Cases (UC)," 2018-09. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/CIM/001_099/002/01.01.01_60/gr_CIM002v010101p.pdf.

[ETS19] "ETSI GS CIM 004 V1.1.1 Context Information Management API," 2019-01. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf.

[ETS6] ETSI GS CIM 006: "Context Information Management (CIM); Information Model (MOD0)".

[Geo12] “OGC GeoSPARQL - A Geographic Query Language for RDF Data” 2012-09.[Online]. Available: https://portal.opengeospatial.org/files/?artifact_id=47664

[HaPa09] Haase P., Palma R., d’Aquin M. D1.1.5 Updated Version of the Networked Ontology Model. Neon project deliverable. February 2009.

[HeSe11] B. Henderson-Sellers. 2011. Bridging metamodels and ontologies in software engineering. J. Syst. Softw. 84, 2 (February 2011), 301–313. DOI:https://doi.org/10.1016/j.jss.2010.10.025

[IDSA20] “International Data Spaces Information Model” 2020-04. [Online]. Available: https://w3id.org/idsa/core.

[IEC19] "IEC White Paper - Semantic Interoperability: challenges in the digital transformation age," 2019. [Online]. Available: https://webstore.iec.ch/publication/65942.

[IOF18] "IOF2020 D3.3 Opportunities and Barriers in the Present Regulatory Situtation for System Development," 2018.

[IvVo11] Ivanov P., Voigt K. (2011) Schema, Ontology and Metamodel Matching - Different, But Indeed the Same?. In: Bellatreche L., Mota Pinto F. (eds) Model and Data Engineering. MEDI 2011. Lecture Notes in Computer Science, vol 6918. Springer, Berlin, Heidelberg

[LeSa] T Lebo, S Sahoo, D McGuinness “PROV-O: The PROV Ontology” Available: https://www.w3.org/TR/prov-o/

[LFr19] L. Frost, "Managing Information using NGSI-LD API," 2019. [Online]. Available: https://oascities.org/wp-content/uploads/2019/01/Managing_Information_ETSI_ISG_CIM_Frostv2_.pdf.

[LoSt] B F Lóscio, E G Stephan, S Purohit “Data on the Web Best Practices: Dataset Usage Vocabulary [Online]. Available: https://www.w3.org/TR/vocab-duv/

[NGS1] "NGSI-LD (Docs) FAQ - Fiware-DataModels," [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_faq/index.html.

[NGS19] "NGSI-LD How to - Fiware-DataModels," [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_howto/index.html.

[ODRL18] “ODRL Vocabulary & Expression 2.2” 2018-02. [Online]. Available: https://www.w3.org/TR/2018/REC-odrl-vocab-20180215/.

[OMV14] OMV 2014. Ontology Definition Metamodel. Version 1.1. OMG standard. https://www.omg.org/spec/ODM

[OnM11] International Organization for Standardization, “ISO 19156:2011 Geographic Information - Observations and measurements”, Geneva

[OtSt19] B Otto, S Steinbuß, A Teuscher, S Lohmann et al. “IDS Reference Architecture Model Version 3.0” 2019-04. International Data Spaces Association [Online]. Available: https://www.internationaldataspaces.org/publications/reference-architecture-model-3-0/

[OWL17] “Time Ontology in OWL” 2017-10 [Online]. Available: https://www.w3.org/TR/owl-time/

[PaLi10] W. Pan and D. Liu, "ORM-ODM: Ontology Definition Metamodel for Object Role Modeling," 2010 3rd

International Conference on Computer Science and Information Technology, Chengdu, 2010, pp. 511-515.

[PaRe16] Palma, Raul & Řezník, Tomáš & Esbrí Palomares, Miguel & Charvat, Karel & Mazurek, Cezary. (2016). An INSPIRE-Based Vocabulary for the Publication of Agricultural Linked Data. 124-133. 10.1007/978-3-319-33245-1_13.

[PROF19] “The Profiles Vocabulary“ 2019 [Online]. Available: https://www.w3.org/TR/dx-prof/

[QUDT] “QUDT CATALOG - Quantities, Units, Dimensions and Data Types Ontologies” 2020-02 [Online]. Available: http://www.qudt.org/2.1/catalog/qudt-catalog.html

[RDF14] "RDF Schema 1.1 W3C Recommendation," 2014. [Online]. Available:https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[Ro15] Robinson, et al. Graph Databases: "New Opportunities for Connected Data". O'Reilly 2nd Edition. Webber, ISBN:1491930896 9781491930892.

[SaKa07] M. Saeki, H. Kaiya. On relationships among models, meta models and ontologies. Procs. 6th OOPSLA Workshop on Domain-Specific Modeling (2007)

[ShaCL17] “Shapes Constraint Language (SHACL)” 2017-07. [Online]. Available: https://www.w3.org/TR/shacl/

[ShaCL20] “IDS Information Model SHACL Integration, GitHub” 2020-04. [Online]. Available:https://github.com/International-Data-Spaces-Association/InformationModel/tree/develop/examples/domain-specific-semantics-using-SHACL

[SSN17] "Semantic Sensor Network Ontology W3C Recommendation 19 October 2017" 2017-12. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[SuGo12] Suarez-Figueroa, M. C., Gomez-Perez, A., & Fernandez-Lopez, M. (2012). The NeOn methodology for ontology engineering. In Ontology engineering in a networked world (pp. 9-34). Springer Berlin Heidelberg.

[Wel] http://www.welchco.com/02/14/60/03/01/1501.HTM

Annex A

This Annex lists the various types of data to be aggregated and shared via DEMETER, as these have by identified by WP5. This is an initial list applicable for the first round of pilots and is expected to be further extended and refined in the course of the project. The types of these data are classified as follows:

- Geospatial data
  - Date / Time
  - Location
  - EGNSS (GPS/EGNOS/Galileo..)
  - Earth Observation
  - Thermal Imagery
  - UAV multispectral imagery
  - PlanetScope hi-res
  - Satellite data (Copernicus)
  - LPIS
  - 3D Point clouds
- Environmental data
  - air temperature
  - air humidity
  - wind speed
  - wind direction
  - solar radiation
  - precipitation
- Water data
  - water salinity
  - water temperature
  - water height
  - PPM
- Soil data
  - soil temperature
  - soil salinity
  - soil humidity
  - soil conductivity
  - soil water tension
  - soil water content
  - Nutrients monitoring/control data
- Crop data
  - Planting details (e.g., date)

  - HyperSpectral Signature
  - Irrigation data
  - Water pumping data
  - NDVI
  - pest info
- Animal Welfare data
  - Beehives data
  - Animal health data
  - Animal respiration data
  - Eating habits data
  - Lameness detection
  - Grazing time
- Product data
  - Product quality data
  - Milk quality data
  - Milk composition data
  - Milk yield
- Machinery data
  - Exhaust temperature
  - NOx-Conversion
  - machine telemetry
  - Engine data
  - After treatment data
- Farm/Logistics data
  - historical farm statistics
  - FADN
  - water consumption
  - energy consumption
  - fuel consumption
  - amount of fertilizer