Document title: Semantic Interoperability and Ontology Design

Version	**Status**	**Date**
1.1	final	2020-12-03

Author	**Contact**
Felix Larrinaga	flarrinaga@mondragon.edu

Semantic Interoperability and Ontology Design

Task 4.2:

Felix Larrinaga
flarrinaga@mondragon.edu
Javier Cuenca
jcuenca@mondragon.edu
Alain Pérez
aperez@mondragon.edu

Abstract

Within the scope of Task 4.2, this document presents the efforts towards providing semantic capabilities for the use cases in the project. The document presents the steps taken to build a layered/modular ontology in the domain of Industry 4.0.

Table of contents

1. Introduction 4
2. Semantic Interoperability 4
3. Architecture: Ontology Application Scenario 6
4. Semantic Scenario Implementation Plan 7
5. Ontology Design and Development. 7
5.1 MODDALS for Ontology Development 8
5.2 MODDALS Steps 8
5.2.1 Preliminary Step: Analysis and Classification of Existing Ontologies 9
5.2.2 Step 1: Ontology Structure Definition 9
5.2.3 Step 2: Domain Knowledge Hierarchy Definition 10
5.2.4 Step 3: Knowledge Classification 12
5.2.5 Step 4: Definition of the Ontology Modular Structure 13
5.3 Implementation of MODDALS in Arrowhead Tools 14
5.3.1 Domains represented by the analysed ontologies 16
5.3.2 Knowledge Hierarchy 17
5.3.2.1 Knowledge hierarchy of the system domain 17
5.3.2.2 Knowledge hierarchy of the manufacturing domain 17
5.3.2.3 Knowledge hierarchy of the asset management domain 18
6. Conclusions 18
7. Appendix A. Application of MODDALS Step 2 in the Industry 4.0 19
7.1 Introduction 19
7.2 Identified knowledge areas 19
7.2.1 Rami 4.0 ontology 19
7.2.2 AutomationML ontology 19
7.2.3 MASON ontology 20
7.2.4 OntoCape ontology 20
7.2.5 SOA ontology 21
7.2.6 Industry 4.0 Knowledge Graph 21
7.2.7 Digital reference ontology 21
7.3 Knowledge area classification 23
7.3.1 Domains represented by the analysed ontologies 23
7.3.2 Knowledge Hierarchy 23
7.3.2.1 Knowledge hierarchy of the system domain 23
7.3.2.2 Knowledge hierarchy of the manufacturing domain 24
7.3.2.3 Knowledge hierarchy of the asset management domain 24
7.4 References 25
8. Revision history 26

8.1 Contributing and reviewing partners....26
8.2 Amendments....26
8.3 Quality assurance....26

1. Introduction

This document presents the efforts in this task (Task 4.2) towards providing semantic capabilities for the use cases in the project. The document presents the steps taken to build a layered/modular ontology in the domain of Industry 4.0.
First, an introduction of semantic interoperability and ontologies is presented.
Next, a typical scenario (architecture) for semantic ontology implementation and validation is outlined.
The actions necessary to implement a solution based on semantic technologies is proposed right after the architecture.
The section continues by addressing the importance of ontologies in a semantic context presenting an ontology modeling approach. As the main contribution in this section, we propose the implementation of a design and development methodology for ontology construction in the context of Industry 4.0 (MODDALS methodology). The first steps in the implementation are presented along with preliminary results.

2. Semantic Interoperability

Interoperability is defined as "*the ability of two or more systems or components to exchange data and use information*" [H. van der Veer and A. Wiles, "Achieving technical interoperability," European Telecommunications Standards Institute, 2008]. Computing systems are distributed and have a dynamic nature. In addition, these systems rely on heterogeneous technologies and use different information representations. These factors hamper the data exchange and data processing between different agents (machines, controllers, sensors …). Therefore, to achieve full interoperability, the exchanged information not only must have a common syntactic base, but also a common structure and common semantics [A. M. Ouksel and A. Sheth, "Semantic interoperability in global information systems," ACM Sigmod Record, vol. 28, no. 1, pp. 5–12, 1999].
Interoperability plays a fundamental role in the context of Industry 4.0. In such a context, the communication between different components such as devices, sensors, production machinery, monitoring and ERP systems requires a certain degree of interoperability to allow the data exchange and data understanding. This is the reason why interoperability is addressed in most reference architectures and reference architectural models for Industry4.0 and the Industrial Internet of Things, such as the Reference Architecture of the Industrial Internet Consortium (IIC) and the Reference Architecture Model Industry 4.0 (RAMI4.0).
There are four types/levels of interoperability according to van der Veer and Wiles:

- **Technical interoperability***:* enables machine-to-machine communications through communication protocols and the hardware/software infrastructure required for those protocols to operate.
- **Syntactical interoperability**: provides a common syntax and encoding to the exchanged data, through data representation languages such as eXtensible Markup Language (XML) or HyperText Markup Language (HTML).
- **Semantic interoperability***:* guarantees that there is a common meaning and understanding of the exchanged data. Specifically, semantic interoperability ensures that IT systems can exchange data in unambiguous ways, through sharing the meaning of data. It is also a foundation for enabling effective machine computable logic (e.g., machine learning algorithms), while boosting inferencing, knowledge extraction and knowledge discovery. Likewise, semantic interoperability facilitates the federation of data and services between information systems, which is particularly useful in scenarios involving interconnected cyber-physical systems.

- **Organisational interoperability***:* Enables the data exchanged between organizations that rely on different infrastructures and heterogeneous information systems. This level of interoperability requires a successful technical, syntactical and semantic interoperability.

Interoperability at semantic level refers to the ability to exchange data between different devices, services and applications in a meaningful way. To achieve this, information regarding the data (metadata) and the working environment is provided along with the data itself. However, the metadata can be provided using a wide range of formats such as JSON, XML, or CSV, hindering the interoperability as different data formats require different ways to process them. In this sense, syntactic and semantic interoperability seems to be similar although there are some dissimilarities. Furthermore, data might be represented in different measurement units or contain other information. The fact of having different data models and schemas might lead to not being able to dynamically inter-operate between devices as they have different descriptions or understandings of resources and operational procedures. In order to achieve semantic interoperability, typically web ontologies are used. Technologies such as Resource Description Framework (RDF), SPARQL, or Web Ontology Language (OWL) are used to reach an agreement on the format and meaning of data by using shared vocabularies regarding the schema. Figure 1 presents alternatives for data extraction, storage and publication using RDF.

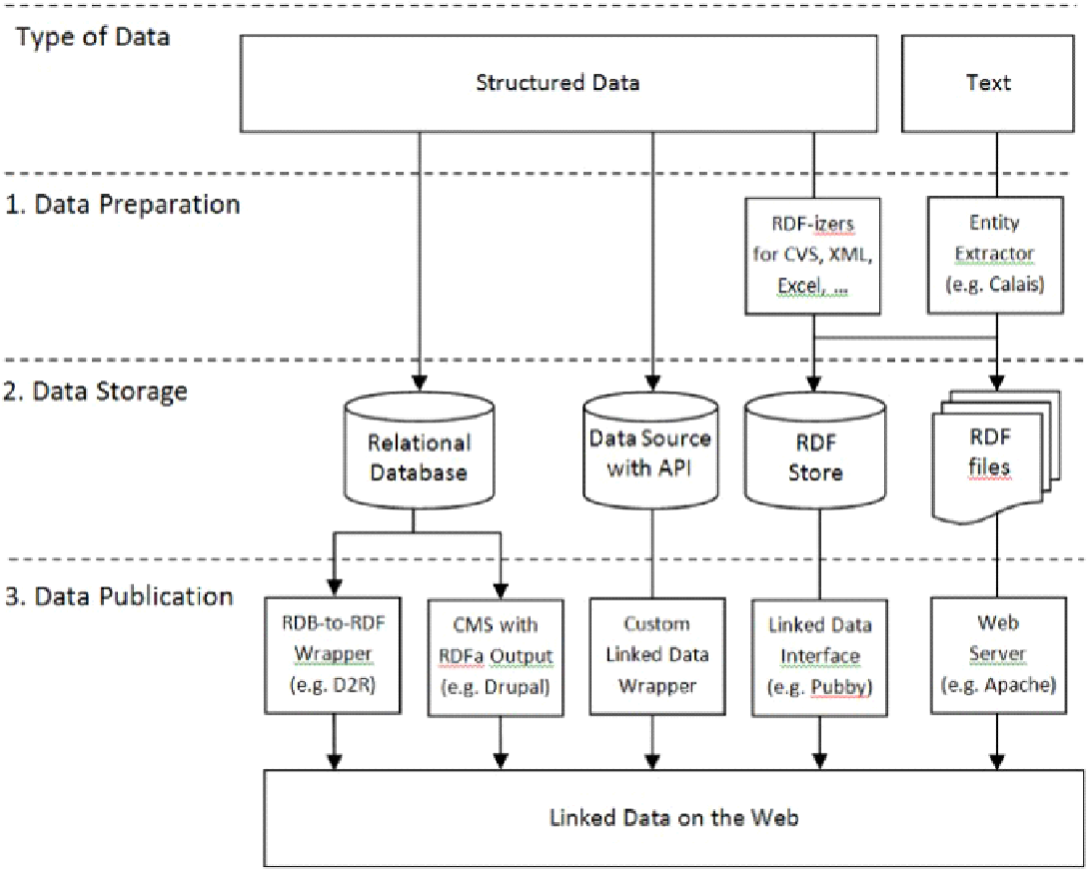

Figure 1: Data extraction, storage and publication using RDF

To model the elements and relations that represent an application domain, ontologies are necessary. Semantic ontologies are formal vocabularies stored as documents on the Web. They describe and represent a data domain as a set of concepts and complex relationships between them. Ontologies enable to create a general knowledge that can be queried, processed and shared across different software applications. Ontologies are developed in OWL (Web Ontology Language) language (https://www.w3.org/OWL/), the standard ontology language proposed by

the W3C (World Wide Web Consortium). OWL is used to represent complex knowledge about things for applications that need to process the content of information instead of just presenting it to humans. OWL provides the basis for creating vocabularies used to describe web data with high expressiveness. With this data representation, intelligent agents can perform advanced data analysis and reasoning for knowledge extraction and decision-making.

3. Architecture: Ontology Application Scenario

The architecture for an ontology application scenario is shown in Figure 2.

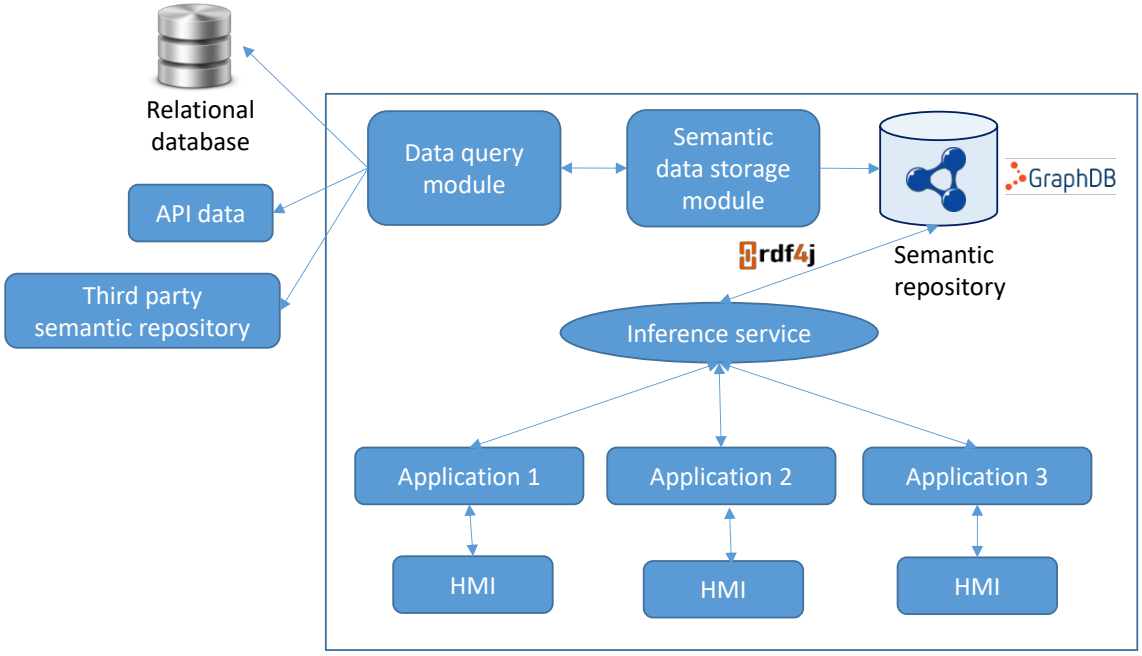

Figure 2: Ontology Application Scenario

Three main elements characterized semantic capabilities in this architecture:
Storage: Data collected and structured according to an ontology needs to be stored for later consumption. Semantic repositories are used to store data represented with an ontology. Semantic repositories are used to store web data in OWL language. All these statements together form a knowledge base. These semantic repositories are similar to database management systems. For example, a technology to build semantic repositories and store the data represented with an ontology is Graph DB (http://graphdb.ontotext.com/).
Reasoning: In addition, the semantic repository includes semantic schemas to automatically reason about the queried data. Many semantic frameworks and repositories use rule-based inference engines, which combine knowledge base assertions and a set of logical rules to infer new information about the knowledgebase.
Semantic applications: Applications that interact with the semantic repository and extract knowledge are also necessary. These applications are the added value offered by the semantic technologies that enable to manage (query, add or remove) data represented with the vocabularies of an ontology. These data are stored usually in the semantic repository. These applications include the libraries of a semantic framework. Semantic frameworks are collections of tools and libraries used to manage semantically represented data. One of these semantic frameworks is RDF4J (http://rdf4j.org/). RDF4J is an open-source semantic web framework that supports RDF data storage, retrieval and analysis. RDF4J libraries enable the management of the semantically represented data. RDF4J framework (formerly SESAME) is written in Java.

The results obtained with applications can be presented using visualization tools. Thus, a Human Machine Interface (HMI) query the data from the semantic repository using the semantic tools and display them.

4. Semantic Scenario Implementation Plan

To build the solution proposed in the previous section, the following steps are necessary:

1) **Design and development of the ontology**. In this first step, the ontology representing the domain is defined and developed. The ontology development must follow the guidelines of a well-known ontology development methodology. The main phases of the ontology development process include the following:
    a. Definition of the ontology requirements.
    b. Analysis of previously developed ontologies that represent the knowledge about automation processes.
    c. Selection and reuse of the elements of a set of the analyzed ontologies.
    d. Implementation of the ontology
    e. Evaluation of the syntax and logical consistency of the ontology.
    f. Publication of the ontology on the web.
2) **Creation of a semantic repository**. The objective of this second step is to store the data represented with the ontology vocabularies. The ontology is loaded into the semantic repository to enable the data storage with the ontology vocabularies.
3) **Development of the semantic application**. The development of the semantic application will follow the following steps:
    a. Definition of the semantic application requirements (depending on the use cases supported by the proposed solutions).
    b. Definition of the methods included by the semantic application.
    c. Implementation of the semantic application using the libraries of the semantic framework (for example RDF4J).
    d. Verification tests of the application against the semantic repository in a local environment.
    e. Development and testing of the HMI.

5. Ontology Design and Development

One of the objectives for this task (Task 4.2) is to provide common data representation elements to improve interoperability among tools and applications. Ontologies are elements that enhance interoperability. When different applications use a common formal vocabulary (ontology) they agree on the data domain and the way it is represented. Consequently, the applications can share data among them. A major drawback in the usage of ontologies is heterogeneity in the representation of a specific domain. That is, two or more ontologies representing the same domain with different terms and relations among them and without providing mappings for those different vocabularies. This is a common situation since in many cases ontology developers do not consider reusing previously existent ontologies or build their ontologies from scratch.
With this in mind, this task plans to develop an ontology to represent the data and the relations in the Industry 4.0 domain considering previously constructed ontologies and providing mapping capabilities between them. The ontology will include the vocabularies that define how the data of the Industry4.0 domain will be represented. The ontology will be design and developed following the steps proposed by the MODDALS methodology [Félix Larrinaga Javier Cuenca and Edward

Curry. Moddals methodology for designing layered ontology structures. *Applied Ontology*, 15(2):185–217, 2020] (see next section). The implementation will be conducted using Protégé ontology editor [https://protege.stanford.edu/] a well-known ontology development tool that enables the creation of ontologies in OWL language.

5.1 MODDALS for Ontology Development

MODDALS presents a methodology that provides guidelines to design a layered structure for reusable and usable ontologies. Layered ontologies classify into different abstraction layers the common domain knowledge (reused by most applications) and the variant domain knowledge (reused by specific application types). The classification of the ontology knowledge into different layers enables ontology developers to reuse only the necessary knowledge at the proper level of abstraction to develop ontologies that satisfy specific application requirements. Hence, the ontology reuse effort in different applications is reduced.
In contrast to previous ontology design methods, MODDALS applies Software Product Lines (SPLs) engineering techniques to systematically (1) identify the ontology common and variant domain knowledge and (2) classify it into different abstraction layers taking as reference existing ontologies. This approach complements domain experts' and ontology engineers' expertise, preventing them from classifying the domain knowledge from scratch facilitating the design of the layered ontology structure.

5.2 MODDALS Steps

This section explains the process followed by MODDALS to design a global ontology. The design process follows four steps (Figure 3), which are described in the following subsections. To explain the different steps we use a previous implementation of the methodology over the energy domain. The ontology produced as a result of the MODDALS methodology implementation in the energy domain is called DABGEO and can be analyzed in http://www.purl.org/dabgeo.

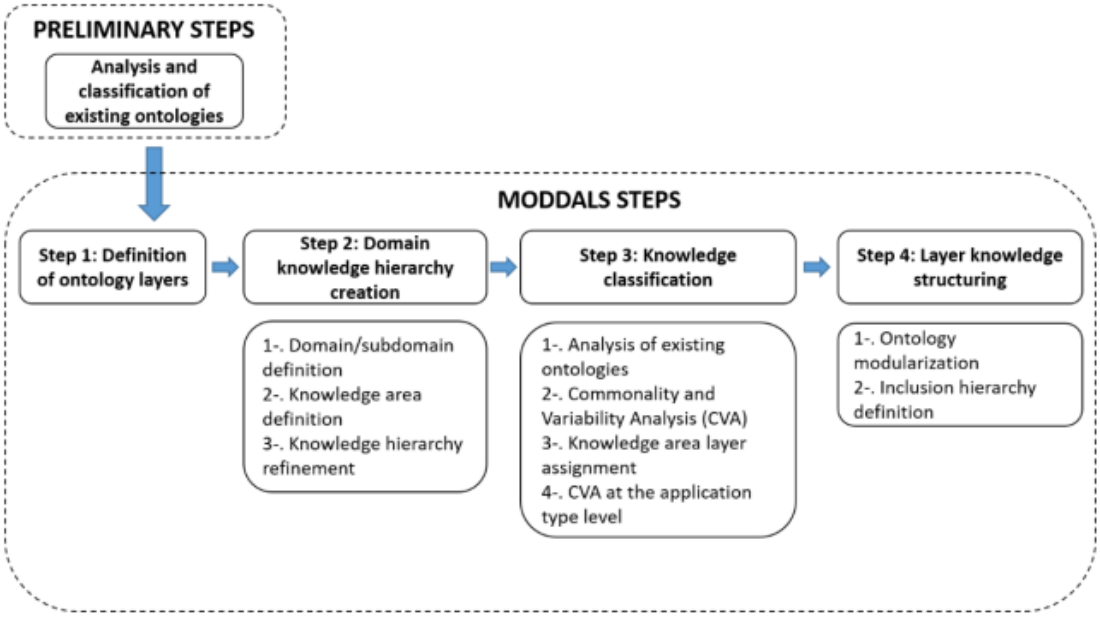

Figure 3: MODDALS Design Steps

5.2.1 Preliminary Step: Analysis and Classification of Existing Ontologies

In this step, domain experts conduct a state of the art of the existing ontologies and the applications they support in the domain concerned. The main objectives of the ontologies and applications are analyzed. The available ontologies that support semantic applications are selected. The ontologies should be as documented as possible, since their knowledge is the input to classify the knowledge in the designed layered structure. The selected ontologies are classified according to the application type they support (assuming that they have been designed and developed in collaboration with domain experts). If already developed ontologies only provide support to specific applications, the domain experts group the applications that perform similar tasks into application types. In the case that the specific applications do not perform similar tasks, each specific application is considered as an application type. It is worth mentioning that if there are only a few ontologies already developed in the domain or these ontologies are reused only by a few application types, the domain analysis will not be representative enough to classify the domain knowledge. The outcome of this step is a classification of existing ontologies according to the application types where they are reused, which is taken as input by the rest of MODDALS steps.

5.2.2 Step 1: Ontology Structure Definition

In this step, a layered structure for the ontology is defined by the domain experts based on the layers proposed by the main ontology design methodologies reviewed in the literature:
J. Morbach, A. Yang, and W. Marquardt, “OntoCAPE:A large-scale ontology for chemical process engineering,” Engineering applications of artificial intelligence, vol. 20, no. 2, pp. 147–161, 2007.
D. Thakker, V. Dimitrova, L. Lau, R. Denaux, S. Karanasios, and F. Yang-Turner, “A priori ontology modularisation in ill-defined domains,” in Proceedings of the 7th International Conference on Semantic Systems, 2011, pp. 167–170.
P. Spyns, Y. Tang, and R. Meersman, “An ontology engineering methodology for DOGMA,” Applied Ontology, vol. 3, no. 1–2, pp. 13–39, 2008.
The structure includes three layers (Figure 4). The *common-domain layer* represents the knowledge common to most scenarios. Variant domain knowledge still common to more than one scenario is included in the *variant-domain layer*. *The domain-task layer* includes the knowledge reused in specific applications and can be further divided depending on the applications used in the domain and their commonality.

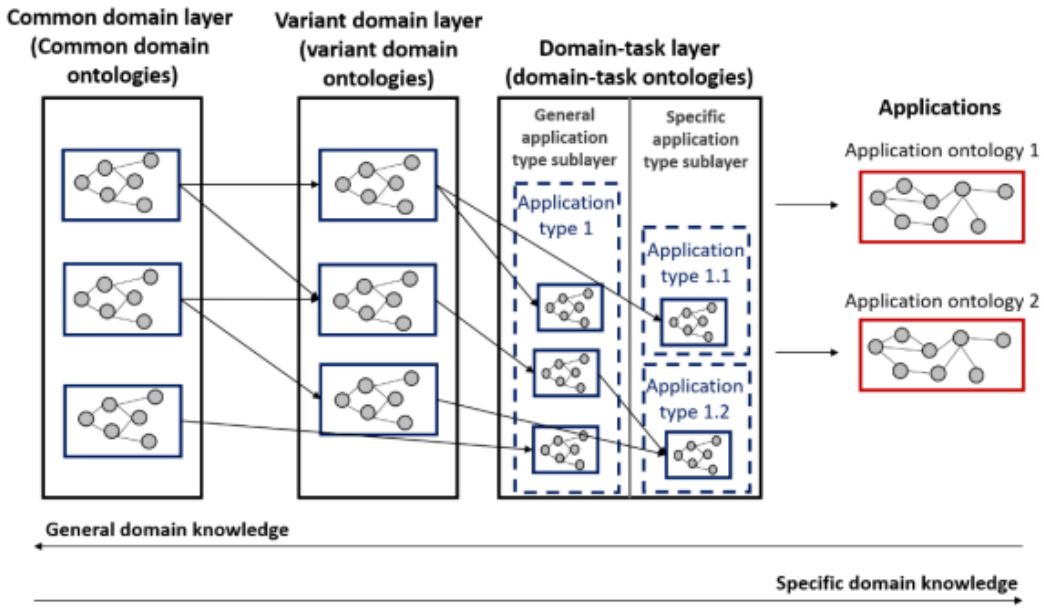

Figure 4: Ontology structure

5.2.3 Step 2: Domain Knowledge Hierarchy Definition

In this step, domain experts and ontology engineers define and structure knowledge.
The knowledge is defined as a knowledge hierarchy in which the represented domains are divided into specific knowledge pieces. This knowledge hierarchy enables (1) to separate the abstract knowledge that is likely to be reused in most applications from the specific knowledge and (2) to classify the defined knowledge pieces into the layers of the ontology structure (performed in Step 3). An example of this hierarchy in another domain (energy) is shown in Figure 5.
This knowledge hierarchy includes three elements:

- *Domains:* the data domains represented by the ontology are located in the first level of the hierarchy. These domains correspond to the ones represented by the existing ontologies in the domain where MODDALS is applied. For instance, one of the domains represented by the energy ontologies taken as reference is the *energy equipment domain*. This domain encompasses the knowledge about energy devices and their operation.
- *Subdomains:* subdomains cover the knowledge of an important part of the domain and are located in the second level of the hierarchy. For instance, the *energy equipment domain* encompasses the *energy consumption systems* and *device operation* subdomains, which represent the knowledge about energy consumption devices and device functional features respectively.
- *Knowledge areas (KAs):* in the third level of the knowledge hierarchy, consider a KA as a potential module of the designed ontology that encompasses the knowledge of a specific topic of a subdomain. For instance, within the *energy consumption systems subdomain* the *appliances KA* represents the knowledge about appliance types. Each KA can be divided into "child" sub-KAs that represent more specific knowledge. For instance, the *appliances KA* includes the *white goods* and *brown goods* KAs, which represent the knowledge about white and brown goods types respectively. Therefore, a sub-KA extends the knowledge of a "parent" KA. Finally, some KAs may require the knowledge from other KAs to represent the knowledge they encompass. For instance, the *energy consumption systems operation KA* describes the states and functionalities of energy consumption systems. This KA requires the knowledge of *device state* and *device functionality* KAs, which represent the knowledge about device states and functionalities respectively.

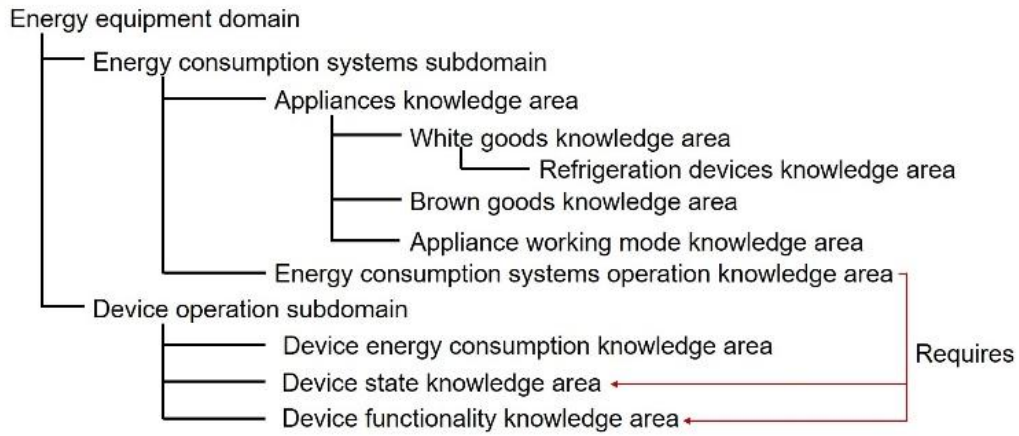

Figure 5: Part of the knowledge hierarchy of DABGEO

As explained later in step 3, the proposed method classifies the ontology domain knowledge based on a *commonality and variability analysis* (CVA) [K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.] of existing energy ontologies. Thus, the knowledge hierarchy of the designed ontology includes the knowledge represented by existing ontologies. The domain experts and ontology engineers collaborate to perform a manual analysis of the elements of existing ontologies in the Protégé ontology editor[1] to identify the domains they represent and to divide them into KAs.

The identified domains are divided into subdomains, which are divided into KAs taking as reference the Competency Questions (CQs) answered by existing ontologies. CQs are the queries that ontologies must answer to ontology-based applications, and they are used to define the ontology functional requirements [M. C. Suárez-Figueroa, “NeOn Methodology for building ontology networks: specification, scheduling and reuse,” Informatica, 2010.]. To answer each CQ the ontology must include a specific part of the represented knowledge. Thus, CQs are a natural guide for splitting the ontology knowledge into KAs

[F. B. Ruy, G. Guizzardi, R. A. Falbo, C. C. Reginato, and V. A. Santos, “From reference ontologies to ontology patterns and back,” *Data & Knowledge Engineering*, 2017.]. CQs offer an abstract method to divide the knowledge represented by existing ontologies regardless of their heterogeneity. Nevertheless, the CQs defined to develop ontologies are not always available.

Ontology engineers analyze manually the elements of existing ontologies (classes, properties and axioms) to identify and extract the CQs they answer. For instance, the existing energy ontologies include the *consumesEnergy*, *actuallyConsumesEnergy* and *maxConsumesEnergy* properties to answer the *What is the energy consumption of a device?*, *How much energy is a device consuming?* and *What is the maximum energy consumption of a device?* CQs respectively. To avoid an unmanageable number of KAs, the CQs covering similar topics were grouped by domain experts to define a KA that encompasses all the knowledge required to answer grouped CQs. For instance, the aforementioned CQs describe knowledge about device energy consumption. They were grouped into the *device energy consumption KA* (it also includes CQs answered by other energy ontologies), which encompasses the knowledge that answers these CQs. The defined KAs are classified into domains and subdomains according to the knowledge they represent and into a hierarchy level according to the knowledge they require or extend.

[1] https://protege.stanford.edu/

Finally, the domain experts provided a complete description of each KA and the knowledge it encompasses.

5.2.4 Step 3: Knowledge Classification

In this step, the ontology engineers classify each defined KA into one layer by applying SPL engineering techniques. Firstly, the existing ontologies are analysed manually with Protégé to determine whether they represent each defined KA. If the ontology contains classes, properties or axioms related with the knowledge encompassed by the KA, the KA is considered as represented. The domain experts collaborate with ontology engineers to give additional explanations about the knowledge encompassed by KAs. It is worth mentioning that if a "child" KA is represented by the ontology, the "parent" KA is also considered as represented.
Secondly, a CVA of existing ontologies is conducted to determine whether the KAs are common among scenarios. In particular, the *application-requirements matrix* technique proposed by Pohl et al. [K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.] is applied (taking as reference application-requirements matrix applied by Moon et al. [M. Moon, K. Yeom, and H. S. Chae, "An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line," *IEEE transactions on software engineering*, vol. 31, no. 7, pp. 551–569, 2005.]) to determine whether the KAs are common to certain scenarios depending on how many ontologies represent them. An application-requirements matrix is used to classify the KAs of each subdomain. As an example, Table 1 shows the application-requirements matrix of a set of KAs of the *energy consumption systems* and *device operation subdomains* (34 KAs were defined in total for these subdomains). The left column contains the KAs of the subdomain. The top rows list the Smart Grid scenarios and the energy ontologies classified by the Smart Grid scenarios they support. We consider a Smart Grid scenario as an energy management application type where energy ontologies are applied.
The matrix indicates if an ontology represents a KA ('X') or not ('-'). With this information, we can deduce which scenarios reuse each KA. We considered that a scenario reuses a KA if the KA is represented by at least one ontology developed to support the scenario. KAs are classified into common and variant according to their *commonality ratio* (CV ratio) (right column in Table 1): the ratio of the number of scenarios that reuse the KA to the total number of scenarios. In particular, a threshold is used as the threshold value of the CV ratio to classify the KAs. The KAs equal or above the threshold are considered as common, while the rest are considered as variant.

	Smart Grid scenarios								
	Smart Home energy management				**Building/district/city energy management**	**Organization energy management**		**Smart Grid Demand Response management**	
Ontologies / Knowledge areas	ThinkHome ontology	EnergyUse ontology	SAREF4EE ontology	Mirabel ontology	SEMANCO ontology	DEFRAM project ontology	DERI Linked dataspace	ProSGV3 ontology	**Commonality Ratio**
Appliances	X	X	X	X	X	X	-	X	100%
Brown goods	X	X	-	-	-	-	-	X	100%
White goods	X	X	X	-	X	-	-	X	75%
Refrigeration devices	X	X	-	-	-	-	-	X	50%
Energy consumption systems operation	X	X	-	-	-	-	-	-	25%
Appliance working mode	-	-	X	-	-	-	-	-	25%

Table 1: Application-requirements matrix

Thirdly, each KA is classified into one layer according to the CVA results. The common KAs are placed in the *common-domain layer*. Variant KAs reused in more than one scenario are assigned to the *variant-domain layer*. The KAs reused only in one scenario are assigned to one of the

sublayers of the *domain-task layer* according to a CVA at the application type level. The KAs reused by more than one application type of a scenario are likely to be reused in more application types of that scenario and are placed in the *scenario sublayer*. The KAs reused only by one application type are assigned to the *application type sublayer*. Following the sample CVA of Table 1 Table 2, the *energy consumption systems operation* and the *appliance working mode* KAs were reused only by Smart Home energy management applications. Thus, they were included in the CVA at application type level. The *energy consumption systems operation KA* was reused by more than one Smart Home energy management application type. Hence, it was placed in the *Smart Grid scenario sublayer*. The *appliance working mode KA* was reused only by one Smart Home energy management applications and placed in the *application type sublayer*.

	Smart Home energy management			
	Home energy assessment	**Home energy saving advice**	**Home appliances Demand Response management**	
Ontologies / Knowledge areas	ThinkHome ontology	EnergyUse ontology	SAREF4EE ontology	Mirabel ontology
Energy consumptions systems operation	X	X	-	-
Appliance working mode	-	-	X	-

Table 2: CVA at application level

5.2.5 Step 4: Definition of the Ontology Modular Structure

In this step, the ontology engineers structure the knowledge of each layer into ontology modules to complete the ontology design. This step is performed taking as reference the ontology modularization principles applied by the main reusable and usable ontology design methods: loosely coupling and self-containment [M. d Aquin, “Modularizing ontologies,” in *Ontology Engineering in a Networked World*, Springer, 2012, pp. 213–233.]. One module is defined for each KA, and placed in one ontology layer/sublayer according to the CVA results. The modules are related according to the knowledge dependencies defined in Step 2. The modules of the *scenario* and *application type* sublayers are classified into the scenario/application types where the corresponding KAs are reused. Figure 6 presents the result of applying MODDALS in the energy domain (part of the final DABGEO ontology)

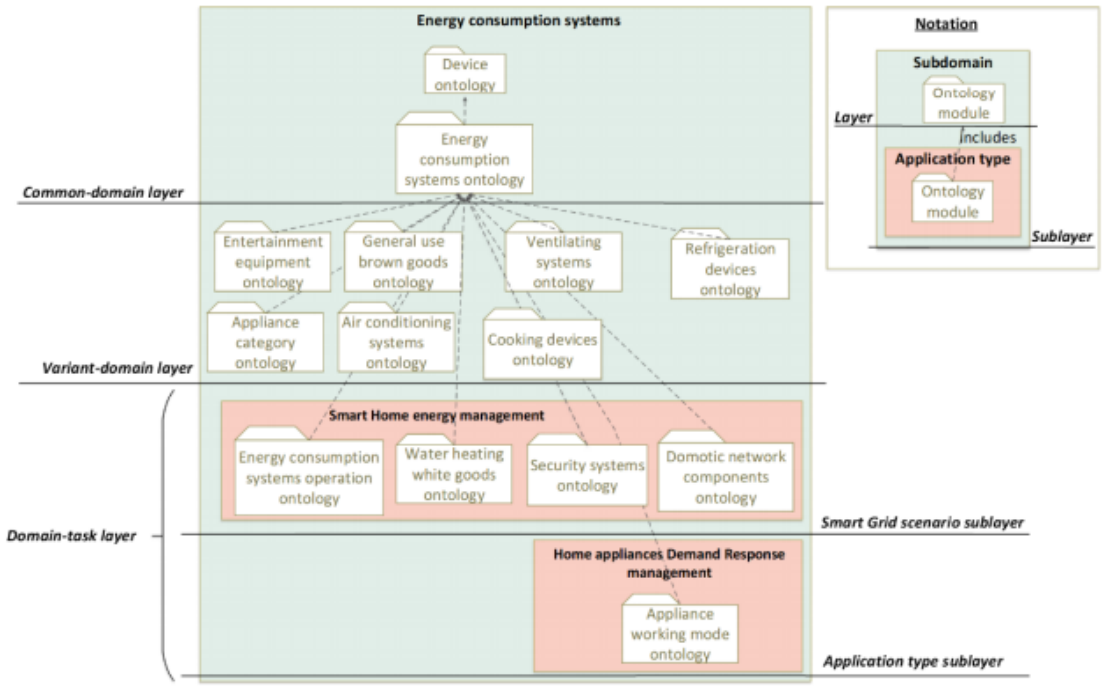

Figure 6: Part of the DABGEO Ontology

5.3 Implementation of MODDALS in Arrowhead Tools

As previously mentioned the idea is to apply MODDALS to build a layered/modular global ontology for the Industry4.0 domain. At the time of writing this deliverable, we have completed the preliminary step, step 1 and step 2. This section presents the actions taken towards the ontology production and the results obtained at this point.
First (preliminary step), we conducted a state of the art for the identification of ontologies related to the domain. Mainly we looked for ontologies in the Industry 4.0 domain identifying which was their general scope or application field and in which format they were available. The following table (Table 3) summarizes the ontologies identified in the field. At this level, we do not evaluate the ontologies.

Ontology	Description	Availabl e format
RAMI ontology	The purpose of the RAMI 4.0 ontology is to be a semantic reference model used by intelligent devices to exchange data, thus enabling a self-organized and resilient product manufacturing process.	.owl
AutomationML ontology	The AutomationML ontology represents the knowledge about the production systems that take part in the manufacturing process such as their function or their role in this process.	.owl
MASON ontology	The MASON ontology is an upper ontology that provides a common representation of manufacturing systems. This ontology is aimed to be the knowledge base of a wide variety of manufacturing systems such as	.owl

	manufacturing cost estimation systems or multi-agent manufacturing systems.	
OntoCAPE ontology	The OntoCAPE ontology is a formal ontology specified for computer-aided process engineering (CAPE). The purpose of OntoCAPE is to be reused by a wide variety of process engineering applications. The ontology has been already reused as a knowledge base for different computer-aided applications for design and implementation process engineering	.owl
SOA ontology	It defines the concepts, terminology, and semantics of SOA in both business and technical terms. The covered terms are systems, the services they offer and the processes they include, events and tasks, as well as information about service contracts.	.owl
Industry 4.0 Knowledge Graph	The graph provides a Linked Data-conform collection of annotated, classified reference guidelines supporting newcomers and experts alike in understanding how to implement Industry 4.0 systems.	
Digital Reference	Extended vocabulary for I4.0 collected in an ontology. See https://atmospheres.research.ltu.se/owncloud/remote.php/webdav/Arrowhead_Tools/WP4%20Tool%20chains/Task%204.2/Work/DataSemanticsCatalogue/O5_%20AHT%20Examples%20of%20Semantic%20Approaches%20v1.0.docx	.rdf/owl
Sequence ontology design pattern	To represent sequence schemas. It defines the notion of transitive and intransitive precedence and their inverses. It can then be used between tasks, processes, time intervals, spatially locate objects, situations, etc.	.owl
Semantic Manufacturing Ontology	See http://i40.semantic-interoperability.org/smo/	
Standards Ontology	The Industry 4.0 Knowledge Graph, I40KG or previously Standards Ontology (STO), represents standards, standardization organizations and standardization frameworks for the Industry 4.0 area. See https://i40-tools.github.io/I40KG/docs/index.html	.rdf
AHT-EP ontology	Ontology for the representation of the engineering phases during the life-cycle of products/solutions.	
GENIAL	Focus on Development Operation integration. See https://atmospheres.research.ltu.se/owncloud/remote.php/webdav/Arrowhead_Tools/WP4%20Tool%20chains/Task%204.2/Work/DataSemanticsCatalogue/O5_%20A	

	HT%20Examples%20of%20Semantic%20Approaches%20v1.0.docx	

Table 3: Industry 4.0 Ontologies

For Step 1, the ontology structure selected is similar to the proposed in previous experiences. See figure for more details on the layers selected for the Industry 4.0 ontology.

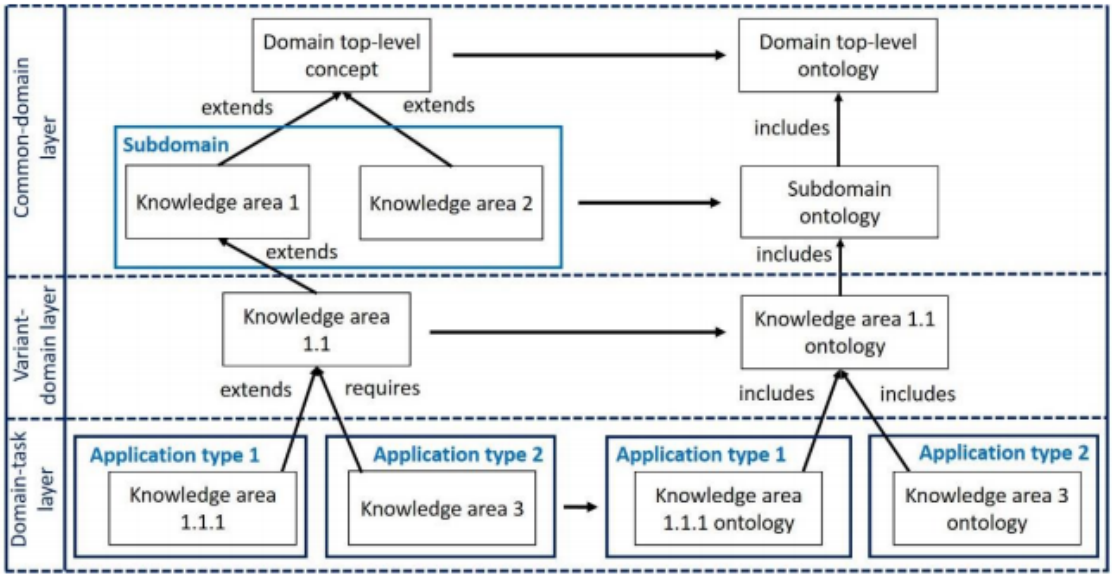

Figure 7: Industry 4.0 Ontology Structure

In Step 2, we analysed in detail the I4.0 ontologies identified in the preliminary step and selected the knowledge areas (and the knowledge they encompass) included by each ontology. We collected that information in an Annex (See document **O5_annex**). The annex presents the knowledge areas identified in each ontology. It also specifies the level of documentation for each ontology, since this is an important aspect to facilitate the comprehension of the represented knowledge.
It is worth mentioning that the knowledge areas were named taking as reference the naming convention applied by the Engineering Village thesaurus (https://www.engineeringvillage.com/home.url). This thesaurus provides the taxonomy and subject classifications used to categorize engineering concepts.
The knowledge areas were classified into the data domains they belong to. Then, the knowledge areas were classified into different abstraction levels and the dependencies between them were defined. This knowledge classification was performed taking as reference how the analysed ontologies relate the knowledge and how these knowledge areas are related according to the Elsevier thesaurus. The result of this classification is presented next.

5.3.1 Domains represented by the analysed ontologies

Taking into account the knowledge areas represented by the analysed ontologies, the ontologies represent three main data domains:

- **System domain**: data about physical or abstract systems, mainly devices. This includes data about device types (i.e., sensors, actuators, communication devices), as well as software systems that provide different services.
- **Manufacturing domain**: data about the actors involved in manufacturing processes in Industry 4.0, i.e., organizations, customers or product suppliers. This includes data about

product suppliers/consumers and organizations, as well as the information about production plants and the products supplied to customers. This domain also encompasses data about manufacturing processes and manufactured products.

- **Asset management domain**: knowledge representation about the Asset Administration Shell and associated concepts (Industry 4.0 objects and sub models that describe them).

5.3.2 Knowledge Hierarchy

In relation to the knowledge hierarchy, the knowledge areas are related through parent-child relations. The "child" knowledge areas include and extend the knowledge of "parent" knowledge areas. Hence, we can say that the knowledge areas placed in the lower levels of the hierarchy include and extend the knowledge of the ones placed in upper levels.
This knowledge classification enables (1) the separation of abstract knowledge that is likely to be reused in most of applications from the specific knowledge and (2) the classification of the defined knowledge pieces into different abstraction levels in the step 3 of MODDALS. In addition, some KAs may represent specific knowledge by combining the knowledge from other KAs. In these cases, the former KAs require the knowledge from the latter. These relations are also reflected in the knowledge hierarchy.
The following subsections show the knowledge hierarchy in each data domain.

5.3.2.1 Knowledge hierarchy of the system domain

The following diagram shows the knowledge hierarchy of the systems domain, including the dependencies between knowledge areas.

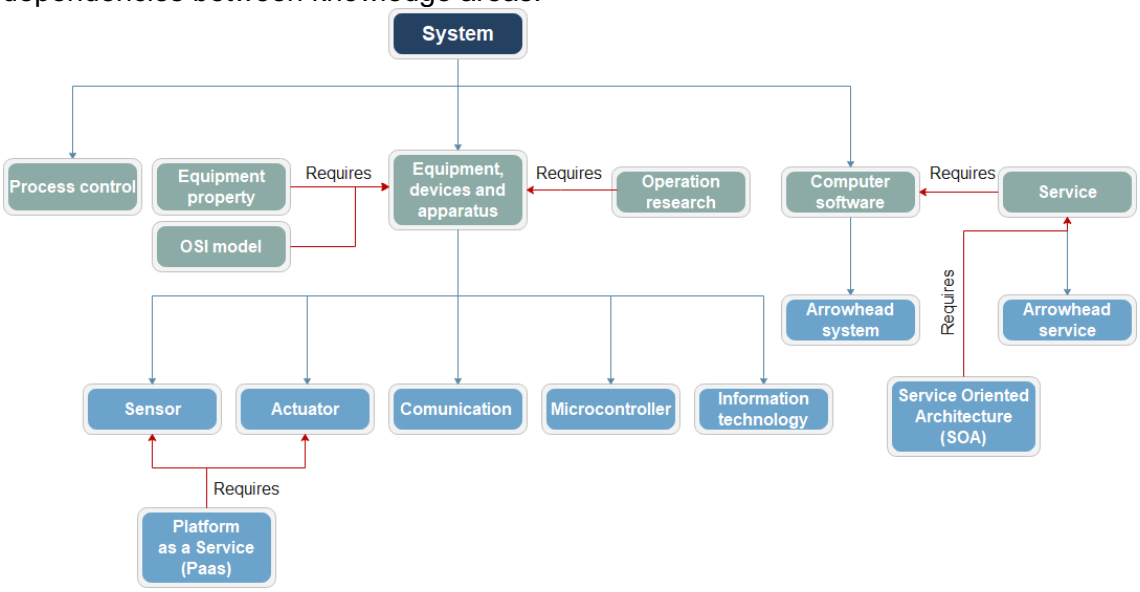

Figure 8: Knowledge hierarchy of the system domain

5.3.2.2 Knowledge hierarchy of the manufacturing domain

The following diagram shows the knowledge hierarchy of the manufacturing domain, including the dependencies between knowledge areas.

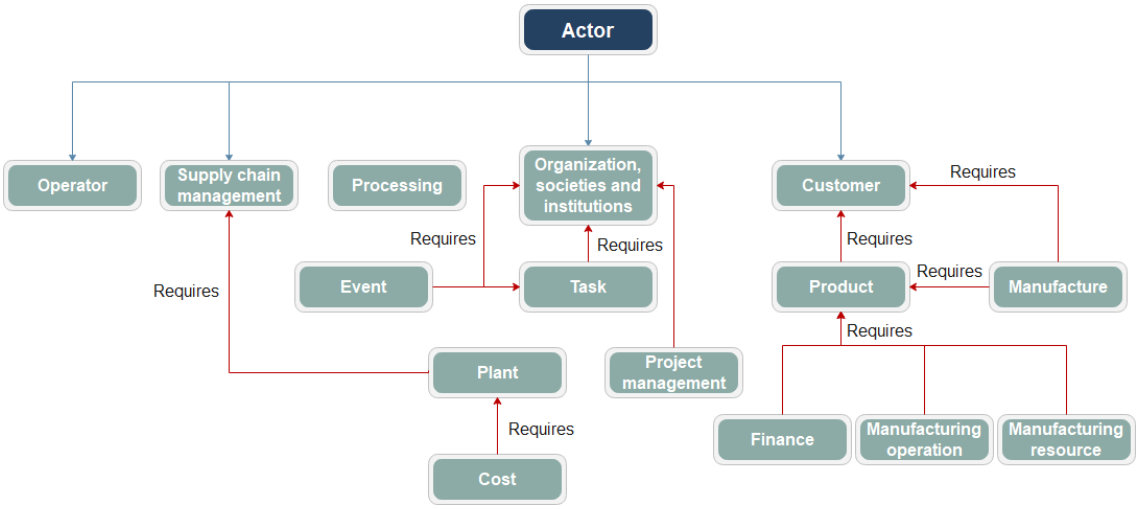

Figure 9: Knowledge hierarchy of the manufacturing domain

5.3.2.3 Knowledge hierarchy of the asset management domain

The following diagram shows the knowledge hierarchy of the systems domain, including the dependencies between knowledge areas.

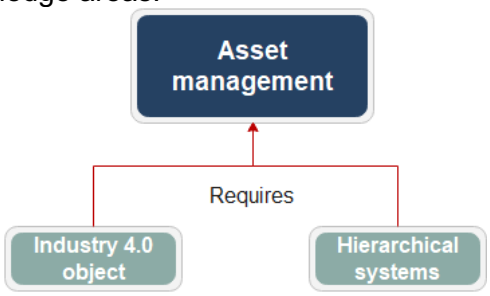

Figure 10: Knowledge hierarchy of the asset management domain

The knowledge hierarchy needs to be fine tune but the project is almost ready to continue with Step 3. It is expected that a first draft of the ontology will be ready by the end of the year.

6. Conclusions

This document proposes an approach to build a layered/modular ontology based on MODDALS methodology for the Industry 4.0 domain. The methodology considers existent ontologies to restructure their knowledge according its commonality. The methodology has been applied previously in other context (energy) and the objective in Arrowhead Tools is to implemented in the context of Industry 4.0.

At this stage we have completed the first steps in the methodology and it is expected to complete its construction by the end of the year. Once the final result is available we will compare this with previous experiences and support partners in the use cases onto the application of the resultant ontology in their environment.

7. Appendix A. Application of MODDALS Step 2 in the Industry 4.0

7.1 Introduction

This document includes the results obtained by applying the step 2 of MODDALS methodology [2] to design the structure of the Industry 4.0 global ontology. Section 2 includes the knowledge areas identified for each of the analysed ontologies. Section 3 includes the classification of the knowledge areas into domains and abstraction levels.

7.2 Identified knowledge areas

This section includes the scope of the analysed I4.0 ontologies and enumerates the knowledge areas (and the knowledge they encompass) included by each ontology. It also specifies the level of documentation of each ontology, since this is an important aspect to facilitate the comprehension of the represented knowledge.
It is worth mentioning that the knowledge areas were named taking as reference the naming applied by the Engineering Village thesaurus [1]. This thesaurus provides the taxonomy and subject classifications used to categorize engineering concepts.

7.2.1 Rami 4.0 ontology

- **Scope:** The purpose of the RAMI 4.0 ontology is to be a semantic reference model used by intelligent devices to exchange data, thus enabling a self-organized and resilient product manufacturing process.

- **Documentation level:** low

- **Represented knowledge areas:**

1. **Asset management:** includes the administration shield concept, the functionalities of the administration shell, assets and sub models that describe an asset.
   - **Industry 4.0 object:** includes data about I4.0 object, such as standards that are used to describe the actual I4.0 Entity/Object, the hierarchy level of an I4.0 object in the RAMI I4.0 model/standard and the application, electrical and engineering data of an I4.0 object.
   - **Sensor:** it represents data about sensors and their output and measurements.
   - **Manufacturing operation:** data about Manufacturing operations and orders, including machining operation as well as control or assembly.

7.2.2 AutomationML ontology

- **Scope:** the AutomationML ontology represents the knowledge about the production systems that take part in the manufacturing process such as their function or their role in this process. The purpose is to support data analysis activities across the discipline/tool boundaries in Production System Engineering (PSE).

- **Documentation level:** high
- **Represented knowledge areas:**
  - **Plant:** data about production plant topology, including the concrete equipment of an actual project – the instance data.

7.2.3 MASON ontology

- **Scope**: The MASON ontology is an upper ontology that provides a common representation of manufacturing systems. This ontology is aimed to be the knowledge base of a wide variety of manufacturing systems such as manufacturing cost estimation systems or multi-agent manufacturing systems.
- **Documentation level:** low
- **Represented knowledge areas:**
  - **Product:** the information about the manufactured product, i.e., geometric features or cost.
  - **Manufacture:** data about Manufacturing operations and orders, including machining operation as well as control or assembly.
  - **Manufacturing resource:** data about the resources (i.e., tools, human resources) used during the manufacturing process.

7.2.4 OntoCape ontology

- **Scope**: the OntoCAPE ontology is a formal ontology specified for computer-aided process engineering (CAPE). The purpose of OntoCAPE is to be reused by a wide variety of process engineering applications. The ontology has been already reused as a knowledge base for different computer-aided applications for design and implementation process engineering.
- **Documentation level:** high
- **Represented knowledge areas:**
  - **System:** data about physical or abstract systems (i.e., technical systems, network systems) and system properties under specific conditions. The systems include complex systems that include subsystems or devices or technical systems developed through an (engineering) design process. The technical system concept may denote all kind of technical artefacts, such as chemical plants, cars, computer systems, or infrastructure systems like a sewage water system.
  - **Plant:** data about production plant topology, including the concrete equipment of an actual project – the instance data.
  - **Process control:** data related with the technical specifications of process control equipment.
  - **Cost:** data about cost models for predicting the (investment) costs of production plants.
  - **Computer software:** data about software applications.

7.2.5 SOA ontology

- **Scope:** it defines the concepts, terminology, and semantics of SOA in both business and technical terms. The covered terms are systems, the services they offer and the processes they include, events and tasks, as well as information about service contracts.

- **Documentation level:** low

- **Represented knowledge areas:**

2. **Task:** data about tasks, actors involved in them.
3. **Event:** data about events and the elements that generate them.
4. **System:** data about physical or abstract systems (i.e., technical systems, network systems) and system properties under specific conditions.

7.2.6 Industry 4.0 Knowledge Graph

- **Scope:** the graph provides a Linked Data-conform collection of annotated, classified reference guidelines supporting newcomers and experts alike in understanding how to implement Industry 4.0 systems.

- **Documentation level:** low

- **Represented knowledge areas:** low, the ontology is being updated.

7.2.7 Digital reference ontology

- **Scope:** the graph provides a Linked Data-conform collection of annotated, classified reference guidelines supporting newcomers and experts alike in understanding how to implement Industry 4.0 systems.

- **Documentation level:** high

- **Represented knowledge areas:**

  - **System:** data about physical or abstract systems (i.e., technical systems, network systems) and system properties under specific conditions.
  - **Event:** data about events and the elements that generate them.
  - **Equipment, devices and apparatus:** data about equipment and machines, as well as their configuration.
  - **Actuator:** data about actuator devices, the actions they perform and the results of these actions.
  - **Communication:** data about devices used for communications, i.e., switch, hub or gateway.
  - **Microcontroller:** data about microcontrollers or logic controllers.
  - **Information technology:** data about office or information Technology (IT) equipment such as computers or printers.
  - **Sensor:** it represents data about sensors and their output and measurements.
  - **Task:** data about tasks, actors involved in them.

- **Processing:** data about business processes and the sub processes/activities they encompass.
- **Manufacture:** data about product orders performed between the product providers and customers.
- **Organization, societies and institutions:** data about organizations, their internal structure (i.e., departments and divisions that compound the organization) and the roles that their departments/units and individuals have.
- **Product:** data about the product supplied by the organization to the customer, manufactured products and their application areas. It also includes data about manufacturing units and their priority class. Each lot is characterized by a product type and a priority class. The priority classes are listed as follows: development, hot, productive, rocket and time coupling.
- **Asset management:** includes the administration shield concept, the functionalities of the administration shell, assets and sub models that describe an asset.
- **Industry 4.0 object:** includes data about I4.0 object, such as standards that are used to describe the actual I4.0 Entity/Object, the hierarchy level of an I4.0 object in the RAMI I4.0 model/standard and the application, electrical and engineering data of an I4.0 object.
- **Customer:** data about the customers of the organization products.
- **Service:** data about services provided by software systems. We consider a service as the information exchanged from a providing software system to a software consuming system.
- **Arrowhead service:** data about Arrowhead compliant services. An arrowhead service is a service that is provided within an Arrowhead environment (i.e., service that is provided by an Arrowhead compliant software system that runs on a device within an Arrowhead local cloud; service is registered at the AH service registry system via its method for service registration and can be discovered and consumed by other Arrowhead compliant software systems.
- **Service Oriented Architecture (SOA):** data about SOA protocols (i.e., REST, SOAP), roles (i.e., service publisher, service subscriber) and service documentation.
- **Computer software:** data about software applications and the functions they perform.
- **Supply chain management:** data about the product supplier.
- **Hierarchical systems:** data about RAMI4.0 elements and hierarchy.
- **Project management:** data about projects and their status.
- **Plant:** data about production plant topology, including the concrete equipment of an actual project – the instance data.
- **Finance:** target costs and revenues associated to delivered products.
- **Operation research:** operations performed by specific equipment and operation modes.
- **Process control:** equipment involved in processes and operations.
- **Operator:** data about users that interact with the equipment only to the degree necessary for the equipment to perform its intended function.
- **Equipment property:** data about equipment configuration or properties such as, on-state voltage or power loss.
- **Arrowhead System:** arrowhead system types and information about these systems (i.e., arrowhead system documentation, services provided by arrowhead systems).
- **Platform as a Service (Paas):** data about entities that host other entities, particularly Sensors, Actuators, Samplers, and other Platforms. A post, buoy, vehicle, ship, aircraft, satellite, cell phone, human or animal may act as platforms for (technical or biological) sensors or actuators.
- **OSI model:** data about the OSI model used to specify the OSI layer of specific devices and OSI protocols/technologies used by specific devices.

7.3 Knowledge area classification

The knowledge areas described in Section 2 were classified into the data domains they belong to. Then, the knowledge areas were classified into different abstraction levels and the dependencies between them were defined. This knowledge classification was performed taking as reference how the analysed ontologies relate the knowledge and how these knowledge areas are related according to the Elsevier thesaurus. This section shows the result of this classification.

7.3.1 Domains represented by the analysed ontologies

Taking into account the knowledge areas represented by the analysed ontologies, the ontologies represent three main data domains:

- **System domain**: data about physical or abstract systems, mainly devices. This includes data about device types (i.e., sensors, actuators, communication devices), as well as software systems that provide different services.
- **Manufacturing domain**: data about the actors involved in manufacturing processes in Industry 4.0, i.e., organizations, customers or product suppliers. This includes data about product suppliers/consumers and organizations, as well as the information about production plants and the products supplied to customers. This domain also encompasses data about manufacturing processes and manufactured products.
- **Asset management domain**: knowledge representation about the Asset Administration Shell and associated concepts (Industry 4.0 objects and sub models that describe them.

7.3.2 Knowledge Hierarchy

The knowledge areas were classified into the data domain they belong. The knowledge areas of each domain were classified into different abstraction levels to create a knowledge hierarchy in each domain.
In the knowledge hierarchy, the knowledge areas are related through parent-child relations. The “child” knowledge areas include and extend the knowledge of “parent” knowledge areas. Hence, we can say that the knowledge areas placed in the lower levels of the hierarchy include and extend the knowledge of the ones placed in upper levels.
This knowledge classification enables (1) the separation of abstract knowledge that is likely to be reused in most of applications from the specific knowledge and (2) the classification of the defined knowledge pieces into different abstraction levels in the step 3 of MODDALS. In addition, some KAs may represent specific knowledge by combining the knowledge from other KAs. In these cases, the former KAs require the knowledge from the latter. These relations are also reflected in the knowledge hierarchy.
The following subsections show the knowledge hierarchy in each data domain.

7.3.2.1 Knowledge hierarchy of the system domain

The following diagram shows the knowledge hierarchy of the systems domain, including the dependencies between knowledge areas.

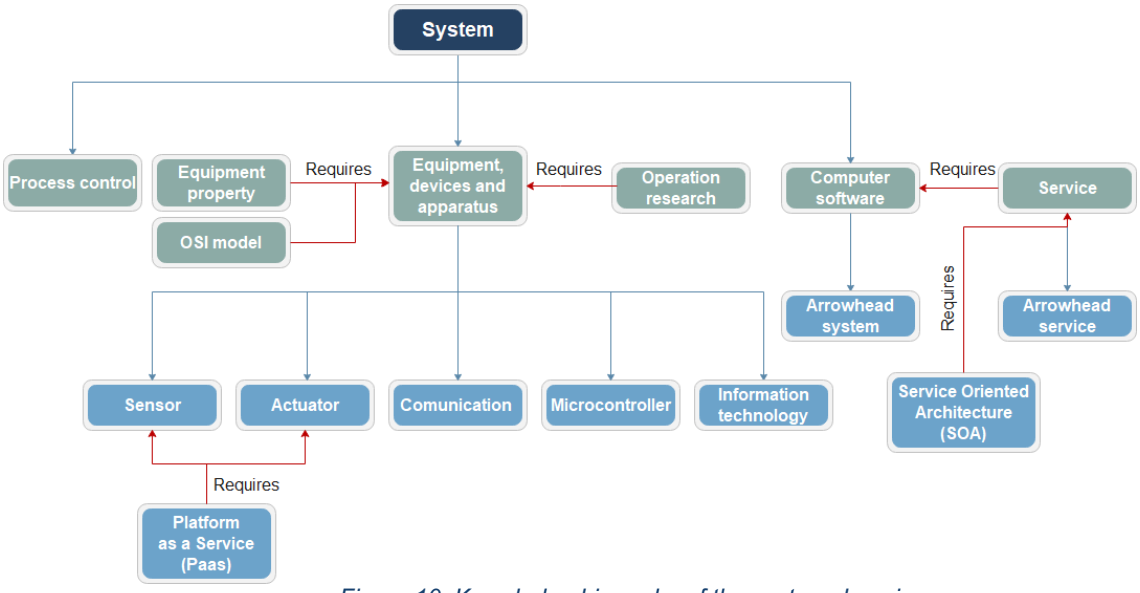

Figure 10: Knowledge hierarchy of the system domain

7.3.2.2 Knowledge hierarchy of the manufacturing domain

The following diagram shows the knowledge hierarchy of the manufacturing domain, including the dependencies between knowledge areas.

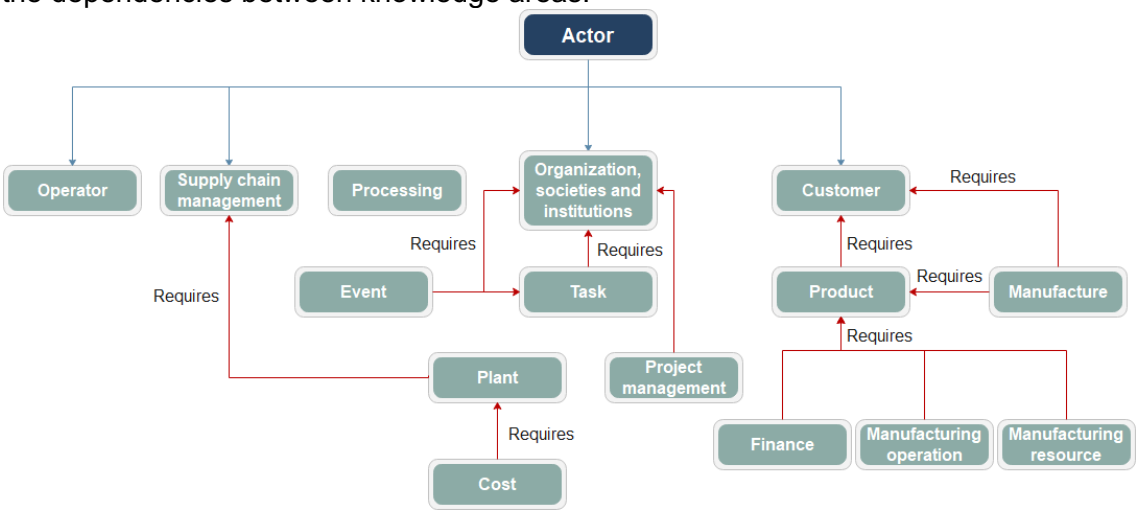

Figure 11: Knowledge hierarchy of the manufacturing domain

7.3.2.3 Knowledge hierarchy of the asset management domain

The following diagram shows the knowledge hierarchy of the systems domain, including the dependencies between knowledge areas.

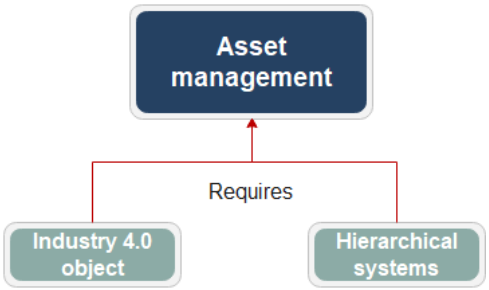

Figure 12: Knowledge hierarchy of the asset management domain

7.4 References

[1] Elsevier. https://www.engineeringvillage.com/search/quick.url, 2020. Accessed: 2020-01-10.
[2] Félix Larrinaga Javier Cuenca and Edward Curry. Moddals methodology for designing layered ontology structures. *Applied Ontology*, 15(2):185–217, 2020

8. Revision history

8.1 Contributing and reviewing partners

Contributions	Reviews	Participants	Representing partner
Initial proposal (draft idea)	Géza Kulcsár, partners in Task 4.2	Javier Cuenca, Alain Pérez and Felix Larrinaga	MGEP
Ontology identification		Javier Cuenca, Alain Pérez and Felix Larrinaga	MGEP
Ontology contribution		Patrick Moder	Infineon
Document version 0.1	Géza Kulcsár	Javier Cuenca, Alain Pérez and Felix Larrinaga	MGEP
Document version 0.2	Géza Kulcsár	Javier Cuenca, Alain Pérez and Felix Larrinaga	MGEP
V1.0	Marek Tatara	Sanity check, formatting adjustments, merging appendix with the document	DAC
V1.0	Jurasky Wiking	Internal Review	IFAG
V1.0	Laurentiu Barna	Internal Review	Wapice
V1.1	Marek Tatara	Addressing comments after the internal review	DAC

8.2 Amendments

No.	Date	Version	Subject of Amendments	Author

8.3 Quality assurance

No	Date	Version	Approved by

1	2020-12-03	1.1	Jerker Delsing