

MODDALS Methodology for Designing Layered Ontology Structures

Javier Cuenca^{**}, Félix Larrinaga^a and Edward Curry^b

^a*Mondragon University/Faculty of Engineering, Loramendi 4, 20500 Arrasate-Mondragon, Spain*
E-mails: jcuenca@mondragon.edu, flarrinaga@mondragon.edu

^b*Insight Centre For Data Analytics, National University of Ireland, Galway, IDA Business Park, Lower Dangan, Galway, Ireland*
E-mail: edward.curry@insight-centre.org

Abstract. Global ontologies include common vocabularies to provide interoperability among different applications. These ontologies require a balance of reusability-usability to minimise the ontology reuse effort in different applications. To achieve such a balance, reusable and usable ontology design methodologies provide guidelines to design and develop layered ontology networks. Layered ontology networks classify into different abstraction layers the domain knowledge relevant to many applications (common domain knowledge) and the domain knowledge relevant only to certain application types (variant domain knowledge). This knowledge classification is performed from scratch by domain experts and ontology engineers. This process is a heavy workload, making it difficult to design the layered structures of reusable and usable global ontologies. Considering how common and variant software features are classified when designing Software Product Lines (SPLs), we argue that SPL engineering techniques can facilitate the domain knowledge classification taking as reference existing ontologies. This paper presents a methodology that provides guidelines to design the layered structure of reusable and usable ontology networks called MODDALS. In contrast to previous methods, MODDALS applies SPL 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. MODDALS methodology is evaluated in the design of the layered structure of a reusable and usable global ontology for the energy domain. The results show that MODDALS enables to classify the domain knowledge taking as reference existing ontologies.

Keywords. Layered ontology networks, methodology, ontology reusability, ontology usability, SPL engineering

1. Introduction

In the context of computer sciences, ontologies are formal vocabularies used to describe and represent a data domain as a set of concepts and relations between them. Ontologies enable to represent a generic knowledge that can be shared across different software applications (Hebeler, Fisher, Blace, & Perez-Lopez, 2011). Some of the main ontology elements are classes (to represent entities), properties (relations used to relate class members), and axioms (restrictions on the properties to express facts about concepts that are always true) (Gruber, 2009).

Ontologies are developed by different engineers, who have different viewpoints when it comes to represent the knowledge of the same data domains. Thus, the creation of ontologies by different developers leads to ontologies that represent the knowledge of the same data domains with different vocabularies. This domain representation diversity, known as *semantic heterogeneity*, leads to an

* Corresponding author: Javier Cuenca, Mondragon University/Faculty of Engineering, Loramendi 4, 20500 Arrasate-Mondragon, Spain. Tel. number: +34 6278598848144. E-mail: jcuenca@mondragon.edu.

1 interoperability problem that hampers the knowledge exchange between knowledge-based applications
2 and hinders the full adoption of ontologies in real scenarios (Maree & Belkhatir, 2015).

3 To date, *global* or *shared* ontologies have been developed in different domains to overcome these
4 interoperability issues, i.e., Soupa (H. Chen, Perich, Finin, & Joshi, 2004). Global ontologies are
5 ontologies that include common vocabularies to provide a common representation and a shared
6 understanding of the domain (Choi, Song, & Han, 2006; Wache et al., 2001). The common knowledge of
7 global ontologies is reused to develop ontologies for different applications (H. Chen et al., 2004; Niknam
8 & Karshenas, 2017). This common knowledge representation overcomes the terminological differences
9 of existing ontologies (the ones that are already developed) in the domain concerned, enabling the
10 knowledge exchange between knowledge bases and applications that use them (Choi et al., 2006; Wache
11 et al., 2001).

12 A global ontology must provide support to different applications in a given domain and must be easily
13 adaptable. That is, it must be *reusable* (Spyns, Tang, & Meersman, 2008). Thus, the ontology must
14 include abstract domain knowledge reused by many applications. However, each application has
15 individual knowledge requirements. If the global ontology is too abstract, the effort of adapting and
16 customizing the knowledge to satisfy specific knowledge requirements would be high. Thus, ontology
17 developers are less likely to reuse the global ontology to develop ontologies for their applications.
18 Considering this, a global ontology must also minimise the ontology reuse effort when it is reused to
19 develop ontologies for specific applications. That is, it must be *usable* (Spyns et al., 2008). Thus, the
20 knowledge of the ontology must be as specific as possible to ease its customisation to specific application
21 requirements. Nevertheless, if the ontology represents the knowledge required by a specific application,
22 the effort of adapting the ontology to applications with different knowledge requirements would be high.

23 With this in mind, both ontology reusability and usability are objectives are “in natural conflict”
24 (Morbach, Wiesner, & Marquardt, 2009), so there is a need to achieve a balance between them (Morbach
25 et al., 2009; Spyns et al., 2008).

26 1.1. Motivation

27 To date, *layered ontology networks* have been applied as the main ontology design approach to achieve
28 a balance of reusability-usability, i.e., OntoCape (Morbach et al., 2009). Layered ontology networks
29 classify into different abstraction layers the *common domain knowledge* (reused by most applications)
30 and the *variant domain knowledge* (reused by specific application types). We consider an *application*
31 *type* a family of applications that perform similar tasks or have similar objectives. Such a classification
32 enables ontology developers to reuse only the necessary knowledge at the proper level of abstraction to
33 develop ontologies that satisfy specific application requirements. Hence, the ontology reuse effort in
34 different applications is reduced (Morbach, Yang, & Marquardt, 2007).

35 Previous works have proposed methodologies to design and develop reusable and usable ontologies
36 that follow the structure of a layered ontology network. These methodologies follow different paths to
37 design and develop the ontologies, but in all of them, the layered structure of the ontology must be
38 designed. The layered ontology structure is an informal model that includes the ontology layers and the
39 knowledge they must include at a conceptual level (as the set of concepts and relations that they must
40 include without going into implementation details) (Morbach et al., 2007).

41 When it comes to design this structure, previous reusable and usable ontology design methodologies
42 provide guidelines to define the ontology abstraction layers and to classify the common and variant
43 domain knowledge into different layers. In these methodologies, the classification of the domain
44 knowledge is performed from scratch based on domain experts’ and ontology engineers’ expertise. They
45 analyse the theoretical framework and the knowledge requirements of the application types that will be
46 supported by the layered ontology network (in collaboration with stakeholders). Based on the gained

1 expertise and the identified knowledge requirements, the ontology knowledge is defined and classified
2 into common and variant (and, by extension, into different layers). Hence, a significant effort is required
3 to classify the ontology knowledge from scratch by applying existing reusable and usable ontology design
4 methodologies. This effort hinders the development of reusable and usable ontologies that represent
5 complex domains and support different applications.

6 In the software engineering field, the main approach to develop reusable and usable software are
7 Software Product Lines (SPLs): software families that contain common reusable parts and variable parts
8 that depend on specific customer needs to support mass customisation (Pohl, Böckle, & Der Linden,
9 2005). For that purpose, software features for a set of applications are analysed and classified into
10 *common features* (common to most applications) and *variant features* (only implemented by specific
11 applications) (Apel, Batory, Kästner, & Saake, 2016; Pohl et al., 2005). The software features of SPLs
12 can be reused to develop new software minimising the effort of adapting the reused software to specific
13 requirements. Thus, layered ontology networks that provide a reusability-usability balance are quite
14 similar in concept to SPLs.

15 When designing SPLs, the software feature classification is performed through a process called *domain*
16 *analysis*. Unlike the design of layered ontology structures, the design of SPLs rarely starts from scratch
17 (Pohl et al., 2005). The domain analysis is usually performed systematically taking as reference the
18 software feature similarities and differences of existing applications and legacy systems (Kang, Cohen,
19 Hess, Novak, & Peterson, 1990; Pohl et al., 2005). Depending on how many applications implement them,
20 the software features are classified into common and variant. This approach makes the SPL design process
21 easier and complements domain experts and software engineers expertise, thus minimising their
22 involvement and effort (Fantechi, Gnesi, John, Lami, & Dörr, 2003; Pohl et al., 2005).

23 After several decades of building semantic web applications in different domains, many developed
24 ontologies are available (Vandenbussche, Ateazing, Poveda-Villalón, & Vatan, 2017). Ontologies are
25 usually developed to be reused and support certain application types. In domains with already developed
26 ontologies, the domain analysis of existing applications applied to design SPLs can be replicated in the
27 ontology engineering field to design the layered structure of reusable and usable ontologies. In particular,
28 the similarities and differences of the knowledge represented by existing ontologies can be analysed to
29 classify the common and variant domain knowledge depending on how many ontologies represent it. This
30 analysis would complement domain experts and ontology engineers' expertise and prevent them from
31 classifying the domain knowledge from scratch.

32 As far as we know, previous reusable and usable ontology design methodologies do not take advantage
33 of existing ontologies to save effort when designing the layered ontology structure (as SPL design
34 approaches do). The design effort reduction is a key enabler of the development of reusable and usable
35 ontologies in complex domains. Therefore, there is the need to define a methodology to design layered
36 ontology structures of reusable and usable ontologies from an existing set of ontologies.

37 Bearing in mind this challenge, the requirements that guide the construction of such methodology are the
38 following:

- 39 1. The main requirement of the proposed methodology is that it should provide techniques to
40 enable the classification of the domain knowledge taking as reference existing ontologies.
- 41 2. In addition, as well as previous reusable and usable ontology design methodologies, the
42 proposed methodology should define precisely the steps conducted to design the layered
43 ontology structure. In particular, the methodology should state clearly the purpose, inputs and
44 outputs, the actors involved, and the techniques to be applied in each step.

1 1.2. Contribution

2 This paper presents the MODDALS (Methodology for Ontology Design based in Domain Analysis
3 and Layered Structure) methodology. MODDALS guides domain experts and ontology engineers to
4 design the layered structure of reusable and usable ontology networks. The output of this process is an
5 informal model with the ontology layers and the knowledge they include at a conceptual level. To define
6 the layered ontology structure, MODDALS applies the main activities and design principles from
7 previous reusable and usable ontology design methodologies (Morbach et al., 2009; Spyns et al., 2008;
8 Thakker et al., 2011).

9 In contrast to these methodologies, MODDALS applies SPL engineering techniques to systematically
10 (1) identify the ontology common and variant domain knowledge and (2) classify it into different
11 abstraction layers taking as reference already implemented ontologies. The knowledge of the ontologies
12 developed for specific application types is usually defined through the collaboration between domain
13 experts and application stakeholders, who translate their knowledge into the ontology (Suárez-Figueroa,
14 2010). In MODDALS, this knowledge is exploited by domain experts and ontology engineers to classify
15 the domain knowledge when designing the layered structure. Therefore, they do not need to analyse the
16 knowledge requirements of different applications and to define and classify the ontology domain
17 knowledge from scratch, facilitating the design of the layered ontology structure.

18 This paper is structured as follows: In Section 2, MODDALS is compared and positioned respecting
19 to previous ontology design and development methods. Section 3 explains the steps in MODDALS.
20 Section 4 shows how MODDALS was applied to design the layered ontology structure of a global
21 ontology for the energy domain. Section 5 presents an empirical evaluation of MODDALS. Section 6
22 summarizes the conclusions of the study as well as future lines of work.

23 2. Related Work

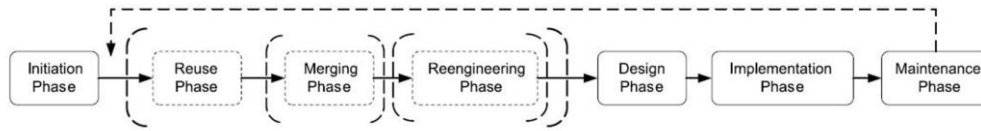
24 This section compares the MODDALS methodology with well-known ontology development
25 methodologies and previous reusable and usable ontology design methodologies. In addition, we indicate
26 when it should and should not be applied.

27 2.1. Ontology Development Methodologies

28 To date a set of well-known ontology development methodologies have been defined, i.e.,
29 METHONTOLOGY (Fernández-López, Gómez-Pérez, & Juristo, 1997), On-to-knowledge (Sure, Staab,
30 & Studer, 2004), DILIGENT (Pinto, Staab, & Tempich, 2004), NeOn (Suárez-Figueroa, 2010) and
31 SABiO (Falbo, 2014). With the exception of NeOn and SABiO, all these methodologies guide to develop
32 ontologies from scratch and do not consider the ontology reuse aspect (Suárez-Figueroa, 2010). NeOn
33 defined different paths to reuse ontologies and to the best of our knowledge, is the methodology that
34 provides more detailed guidelines when reusing ontologies. As well as NeOn, SABiO also supports
35 ontology reuse, with the difference that SABiO is thought to develop both domain and operational
36 ontologies (Falbo, 2014).

37 MODDALS takes as reference the knowledge of existing ontologies to design the layered ontology
38 structure. Hence, once the structure is designed, the knowledge of existing ontologies will be reused to
39 implement the layered ontology network. Therefore, MODDALS fits better with and can be applied as
40 an internal step of NeOn. NeOn defines a set of flexible scenarios to develop ontologies and ontology
41 networks. These scenarios correspond to the methods (i.e., reuse, reuse and merge) that can be applied to
42 reuse existing knowledge sources (i.e., existing ontologies or non-ontological resources) to develop

1 ontologies. Fig 1, summarizes the different phases that the ontology development process can follow
 2 depending on the selected scenario (to see more detailed information about each phase, we refer the reader
 3 to (Suárez-Figueroa, 2010)).

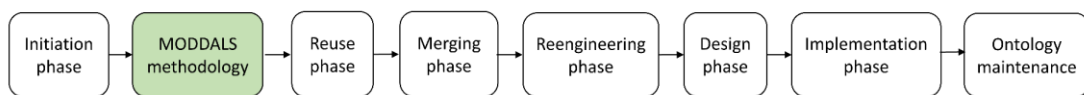


4
 5 Fig. 1: Ontology network life-cycle models proposed in the NeOn Methodology framework (Suárez-Figueroa, Gómez-Pérez, & Fernández-
 6 López, 2015)

7
 8 Since MODDALS classifies the ontology domain knowledge taking as reference existing ontologies,
 9 the knowledge of the designed layered structure includes the knowledge from these ontologies. Existing
 10 ontologies are analysed to classify this knowledge into different layers. The output of this process is an
 11 informal model of the ontology that contains the ontology layers and the knowledge they include. Hence,
 12 within the ontology life-cycle, MODDALS covers part of the ontology reuse process. In particular, it
 13 proposes a new scenario for reusing ontologies: organisation of the various existing ontologies into an
 14 overall layered ontology structure.

15 In addition, in MODDALS the knowledge that the ontology must represent is defined (the knowledge
 16 from existing ontologies). In contrast, in NeOn the knowledge of the ontology is defined from scratch as
 17 the functional requirements of the ontology during the *ontology initiation* phase. Therefore, MODDALS
 18 covers part of this phase.

19 Considering the ontology development phases covered by MODDALS, it should be applied right after
 20 the *ontology initiation* phase and before the *ontology reuse* phase of NeOn (Fig. 2). During the ontology
 21 initiation phase, the ontology purpose, scope and non-functional requirements should be defined. Then,
 22 MODDALS should be applied to (1) search for existing ontologies in the domain concerned, (2) define
 23 the ontology knowledge and (3) define the layered ontology structure. Then, in the ontology reuse phase,
 24 the existing ontologies should be reused so that the developed ontology represents the defined knowledge
 25 according to the defined layered structure.
 26



27
 28 Fig. 2: Application of MODDALS within NeOn methodology phases

29
 30 They describe and represent a data domain as a set of concepts and relationships between them to create
 31 a generic knowledge that can be shared across different software applications.

32 **2.2. Ontology Classification Frameworks and Layered Ontologies**

33 The first proposals applied to design ontologies that provide a balance of reusability-usability
 34 correspond to frameworks that classify ontologies according to their generality/specificity level. Guarino
 35 (1997) presented the first ontology classification framework, which distinguishes between the following
 36 ontologies:
 37

- 1 • *Upper ontologies*: they represent general and domain independent knowledge and concepts (i.e.,
2 object, state) that can be reused in different domains.
- 3 • *Domain ontologies*: they extend the knowledge of the upper ontologies, since they represent the
4 knowledge of a particular domain. Some domain ontologies represent only domain top-level
5 knowledge, whereas other domain ontologies include domain-specific knowledge. Thus, some
6 domain ontologies can extend the knowledge of other domain ontologies.
- 7 • *Task ontologies*: they extend the knowledge of domain ontologies and represent the knowledge
8 related to generic tasks or activities. Thus, these ontologies are reused by applications of any
9 domain that perform similar tasks.
- 10 • *Application ontologies*: they are the ontologies that include the most specific knowledge, since
11 they represent the knowledge reused by certain applications.

12 This classification was refined later by Gómez-Pérez, Fernández-López, & Corcho (2006), who
13 introduced *domain-task ontologies*. These ontologies represent the domain knowledge related to tasks
14 performed by applications of a given domain. Hence, these ontologies represent the domain knowledge
15 reused by certain application types within a specific domain and they are located between domain and
16 application ontologies.

17 The main methods focused on improving the ontology reusability-usability balance deal with designing
18 layered ontology networks based on previous ontology classification frameworks. Layered ontology
19 networks classify represented domain knowledge in different abstraction layers according to their
20 knowledge generality/specificity level, thus separating the common and variant domain knowledge
21 (Morbach et al., 2009; Thakker et al., 2011). The knowledge of each layer is classified into ontology
22 modules that represent the knowledge of a particular topic of the represented domain (D'Aquin, 2012).

23 An example of the structure of a layered ontology network is shown in Fig 3. Top-level layers include
24 upper ontologies to represent general knowledge. Low-level layers include domain and domain-task
25 ontologies to represent the common and variant knowledge about represented domains. The lower the
26 layer is, the more specific concepts and relations it includes. Within this layered structure, some ontology
27 modules extend the knowledge of other modules, since they represent more specific concepts and
28 relations. The ontology modules include the knowledge of the ontology modules they extend. These
29 ontologies are reused, adapted and combined by ontology engineers to develop application ontologies
30 that fit application-specific knowledge requirements. With the layered structure, ontology developers can
31 analyse and select at the proper level of abstraction the necessary knowledge to develop application
32 ontologies (Morbach et al., 2007). Hence, the ontology reuse effort in different applications is reduced.
33

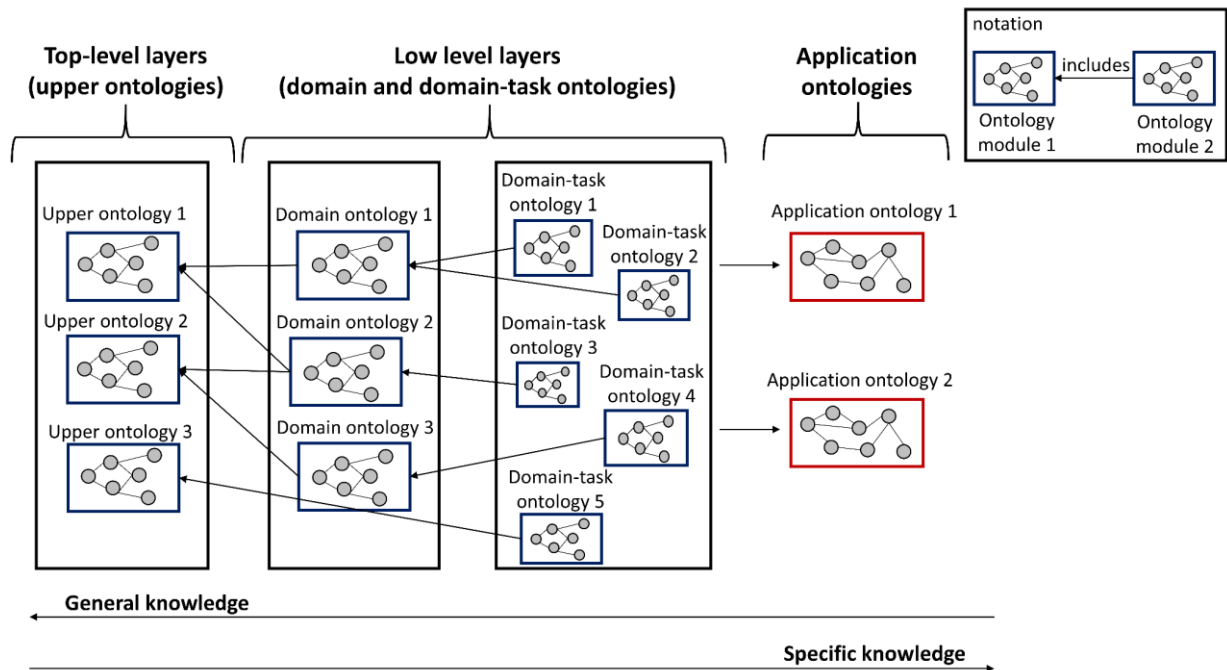


Fig. 3: Sample structure of a layered ontology network

2.3. Reusable and Usable Ontology Design Methods

In the last decade, Ontology Design Patterns (ODPs) have been researched as the main solution for improving ontology reusability (Gangemi, 2005; Hitzler et al., 2016). ODPs are small ontologies that represent domain independent knowledge and act as ontology building blocks to improve ontology reusability. In contrast to ODPs, MODDALS is focused on designing the layered ontology structure to represent only the domain knowledge of the ontology. Therefore, it is applicable to design the low-level layers of the layered ontology structure. The knowledge represented by ODPs is more abstract and would be located in upper layers within a layered ontology network. Hence, MODDALS is complementary to ODPs.

On the other hand, in the last decade several methodologies have been proposed to design and develop reusable and usable ontologies that follow the structure of a layered ontology network. These methodologies follow different paths to design and develop the ontologies but in all of them, the layered structure of the ontology must be designed.

Spyns et al. (2008) presented the DOGMA methodology, which is based on the DOGMA framework (Jarrar & Meersman, 2008). The DOGMA framework specifies how to represent and separate the common and variant domain knowledge within a reusable and usable ontology. Thakker et al. (2011) set out a methodology to develop reusable and usable ontologies that represent ill-defined and complex domains. This methodology proposes a set of ontology layers to classify the common and variant domain knowledge and explains which knowledge should be included in each layer. In contrast to previous approaches, Morbach et al. (2009) developed the OntoCape ontology, a highly reusable and usable ontology for the chemical process engineering domain. Morbach et al. (Morbach et al., 2007) detail the OntoCape ontology design and implementation methodology and process.

When it comes to design the layered ontology structure, the main activities conducted by previous methodologies are the following: (1) define the ontology abstraction layers and the kind of knowledge

1 they will include (common or variant), (2) define the ontology knowledge, (3) classify the common and
2 variant domain knowledge into different layers and (4) structure the knowledge in each layer. The
3 classification of the domain knowledge is performed from scratch based on domain experts' and ontology
4 engineers' expertise.

5 In contrast to these methods, MODDALS provides guidelines to classify the domain knowledge based
6 on a domain analysis of existing ontologies applying SPL engineering techniques. MODDALS also has
7 common aspects with previous reusable and usable ontology design methodologies. MODDALS applies
8 the main activities and ontology design principles applied by these methodologies. Therefore, the purpose
9 of MODDALS is not to substitute these methodologies to improve the domain knowledge classification.
10 It offers an alternative method to classify the common and variant domain knowledge.

11 2.4. MODDALS Usage

12 Bearing in mind the features of MODDALS and its position with respect to the previous works, it
13 should be applied when the following conditions are met:

- 14 1. The developed layered ontology network must provide a balance of reusability-usability, since it
15 is developed to be reused by different applications in a given domain.
- 16 2. There are already developed ontologies that support different application types in the domain.
- 17 3. The developed layered ontology network will be applied in a complex domain.
- 18 4. The developed layered ontology network will represent domain knowledge.

19
20 Otherwise, it should not be applied in the following cases:

- 21 1. The layered ontology network is developed for a specific application.
- 22 2. There are no ontologies developed in the domain concerned.
- 23 3. The domain where the developed layered ontology network is applied is not complex.
- 24 4. The developed layered ontology network will no represent domain independent knowledge.

26 3. MODDALS Methodology

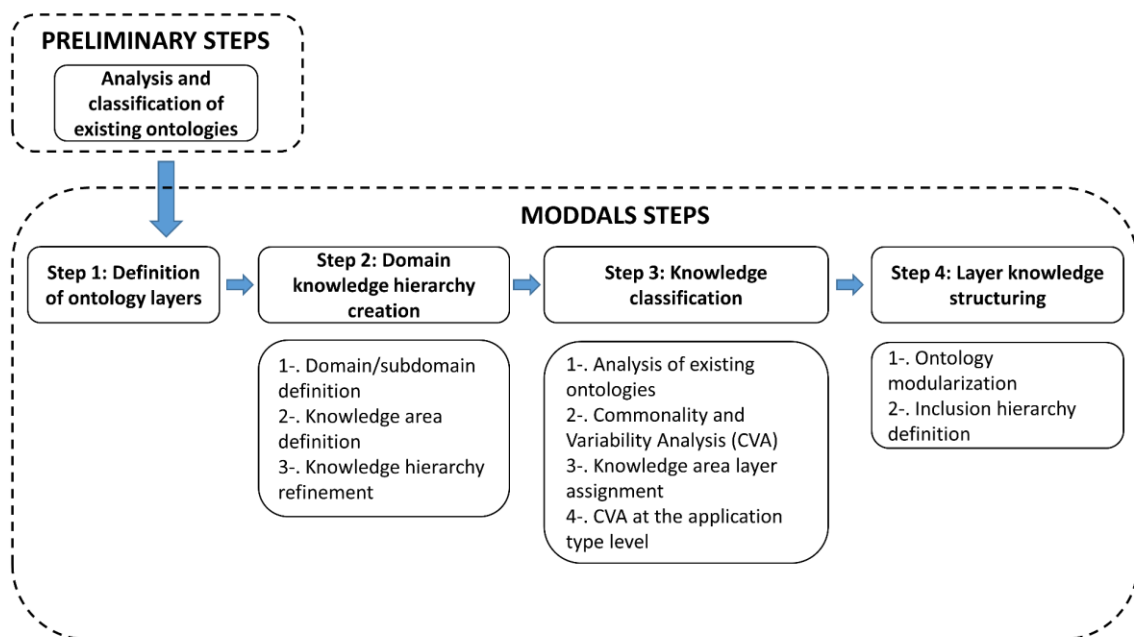
27 This section explains the steps in MODDALS, which were defined bearing in mind the requirements
28 defined in Section 1.1. Considering these requirements, MODDALS takes as input previous reusable and
29 usable ontology design methodologies (Morbach et al., 2009; Spyns et al., 2008; Thakker et al., 2011)
30 and well-known SPL engineering techniques (Moon, Yeom, & Chae, 2005; Pohl et al., 2005).

31 On the one hand, as well as previous reusable and usable ontology design methodologies, MODDALS
32 should define precisely the steps conducted to design the layered ontology structure. Considering this
33 requirement, MODDALS steps have been defined bearing in mind the main activities applied by previous
34 reusable and usable ontology design approaches (Morbach et al., 2009; Spyns et al., 2008; Thakker et al.,
35 2011): (1) definition of the ontology abstraction layers and the kind of knowledge will include (common
36 or variant), (2) definition of the ontology knowledge, (3) classification of the common and variant domain
37 knowledge into different layers and (4) structure the knowledge in each layer.

38 On the other hand, the main requirement of MODDALS is to provide techniques to enable the
39 classification of the domain knowledge taking as reference existing ontologies. Considering this
40 requirement, the activities applied to design the layered ontology structure were adapted so that
41 MODDALS classifies the domain knowledge taking as reference existing ontologies. As stated in Section
42 1.1, SPL design techniques enable to classify software features taking as reference the similarities and

1 differences of existing applications. Therefore, MODDALS applies well-known SPL design techniques
 2 (Moon et al., 2005; Pohl et al., 2005) to classify systematically the domain knowledge.

3 Based on the aforementioned activities and techniques, MODDALS encompasses four main steps.
 4 These steps involve the collaboration between domain experts and ontology engineers and are conducted
 5 sequentially. In addition, MODDALS takes as reference already developed ontologies to classify the
 6 domain knowledge into different abstraction layers. Therefore, before applying the MODDALS steps, a
 7 preliminary step is required: *analysis and classification of existing ontologies*. Once the exiting ontologies
 8 have been selected and analysed, the methodology itself is implemented (Fig. 4).
 9



10
 11 Fig. 4: MODDALS methodology steps

12 **3.1. Preliminary Step: Analysis and Classification of Existing Ontologies**

13 In this step, domain experts conduct a state of the art of the existing ontologies and the applications
 14 they support in the domain concerned.

15 The main objectives of the ontologies and applications are analysed. The available ontologies that
 16 support analysed applications are selected. The ontologies should be as documented as possible, since
 17 their knowledge is the input to classify the knowledge in the designed layered structure. The selected
 18 ontologies are classified according to the application type they support (assuming that they have been
 19 designed and developed in collaboration with domain experts). If already developed ontologies only
 20 provide support to specific applications, the domain experts group the applications that perform similar
 21 tasks into application types. In the case that the specific applications do not perform similar tasks, each
 22 specific application is considered as an application type.

23 It is worth mentioning that if there are only a few ontologies already developed in the domain or these
 24 ontologies are reused only by a few application types, the domain analysis will not be representative
 25 enough to classify the domain knowledge, as well as occurs when designing SPLs (Kang et al., 1990).
 26 Therefore, MODDALS is not applicable in these cases. To define the minimum sample of ontologies to
 27 apply the methodology the Feature-Oriented Domain Analysis (FODA) model (Kang et al., 1990) is taken

1 as reference, since it establishes the main principles and the main steps of the SPL domain analysis
2 process (L. Chen, Ali Babar, & Ali, 2009). According to the FODA model, a domain analysis must take
3 as input at least three applications (as divergent in functionality as possible). Therefore, we consider
4 ontologies that provide support to at least three application types must be already developed within the
5 domain where MODDALS is applied as a minimum sample to apply the methodology. If these conditions
6 are not met, one of the reusable and usable ontology design methods introduced in Section 2.3 should be
7 applied to design the ontology structure.

8 The outcome of this step is a classification of existing ontologies according to the application types
9 where they are reused, which is taken as input by the rest of MODDALS steps.

10 3.2. Step 1: Definition of Ontology Layers

11 In the first step, domain experts define the layers of the ontology network that classify the domain
12 knowledge and the kind of knowledge they include.

13 The layered structure proposed by MODDALS has been defined taking as reference the layers proposed
14 by the previous reusable and usable ontology design methodologies. In addition, the defined layers must
15 be compatible and comply with the knowledge classification method proposed in MODDALS: a domain
16 analysis of existing ontologies by applying SPL engineering techniques.

17 When it comes to represent the domain knowledge, all the reusable and usable ontology design
18 methodologies reviewed in Section 2.3 propose (1) a layer that includes the common domain knowledge
19 reused by all application types covered by the layered ontology network and (2) a layer that includes the
20 variant domain knowledge reused by specific application types. A set of application types in a given
21 domain will have knowledge in common, while each application will require specific knowledge (Spyns
22 et al., 2008). Hence, the aforementioned layers are mandatory in a layered structure. These layers are
23 compatible with the knowledge classification method applied on MODDALS, since the domain analysis
24 classifies the software features (in this case knowledge) into the ones common to all applications and
25 those that are implemented by specific applications (Pohl et al., 2005).

26 In SPL design, there is no a middle ground when classifying the software features, since they are usually
27 implemented by most of applications or specific applications (Moon et al., 2005). However, in
28 MODDALS we apply the domain analysis to classify knowledge instead of software features. Depending
29 on the knowledge similarities and differences of existing ontologies, there might be knowledge that is not
30 common but still reusable across a set of application types. Therefore, the layered ontology network must
31 include an intermediate layer. In these sense, the OntoCape ontology (Morbach et al., 2009) adds a layer
32 that contains the domain knowledge not common but still relevant to several application types.

33 Considering these aspects, we propose in MODDALS a layered-structure that combines the layers
34 proposed by previous approaches and contains three layers (Fig. 5). These layers constitute a template
35 where the ontology knowledge is classified in the next steps. Previous reusable and usable ontology
36 design methods do not follow a pre-established standard to name the layers. They name differently the
37 layers that contain the same kind of knowledge. Hence, we have defined the name of the layers based on
38 the kind of knowledge (common knowledge, variant knowledge still common to more than one
39 application type, variant knowledge only reused by specific application types) they include.

- 41 • The *common-domain layer* includes domain ontologies that represent the top-level knowledge of
42 each domain. The domain ontology modules of this layer also represent the common domain
43 knowledge. The knowledge in this layer is extended by the knowledge in the next two layers,
44 which are more specific.
- 45 • The *variant-domain layer* includes domain ontologies that represent the variant domain
46 knowledge still common to more than one application type.

- The *domain-task layer* includes domain-task ontologies that represent the variant domain knowledge reused by specific application types. The ontology modules of this layer are classified according to the application type where they are reused. Thus, the structure of this layer can vary depending on the application types supported by the layered ontology network. MODDALS classifies the domain knowledge taking as reference existing ontologies. Thus, only the application types supported by existing ontologies are taken as reference to define the ontology structure of this layer. Possible future application types are not taken into account since “*a complete domain theory is lacking in almost any complex (engineering) domain*” (Morbach et al., 2007).

In some domains, a set of applications that belong to an application type can be grouped into a more specific application type, since they have specific objectives in common. In these cases, the *domain-task layer* is divided into two sublayers. The sublayers separate the knowledge reused only by a specific application type from the knowledge still relevant for more specific application types encompassed by the general application type. For instance, let us consider that the *application type 1* encompasses the *application type 1.1* and the *application type 1.2*. The knowledge reused by both *application type 1.1* and *application type 1.2* could be relevant for any other application type encompassed by the *application type 1*. This knowledge is placed in the *general application type sublayer*. In contrast, the knowledge reused only by the *application type 1.1* is only relevant for that application type. Therefore, this knowledge is separated from the one relevant for both application types. This knowledge should be placed in the *specific application type sublayer*. The domain experts can also name each sublayer using the terms in the domain concerned to facilitate the distinction between the two sublayers (as done in Section 4, where MODDALS is applied in the energy domain).

The outcome of this step is a high-level structure of the layered ontology network with the layers described above.

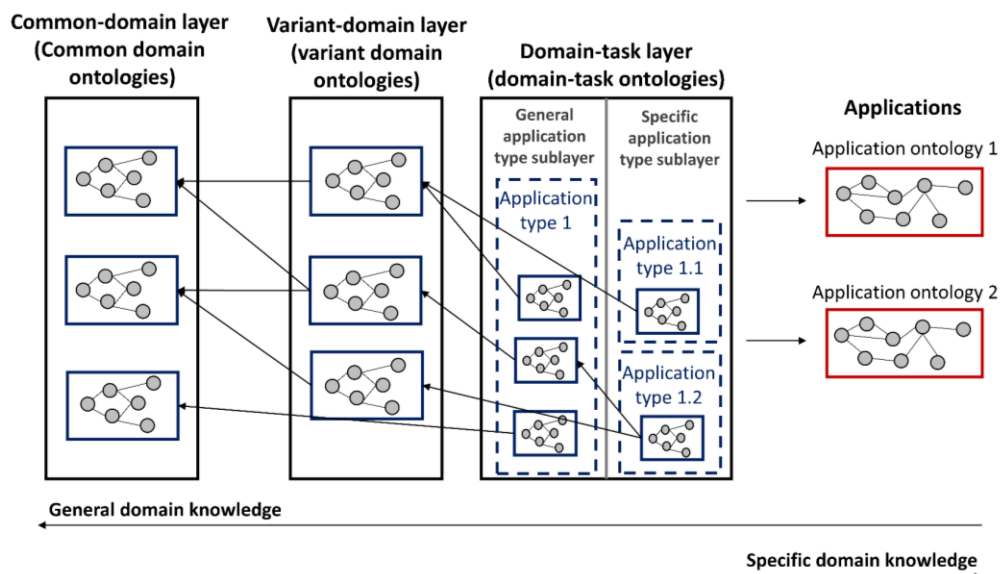


Fig. 5: Ontology structure proposed by MODDALS

1 3.3. Step 2: Domain Knowledge Hierarchy Creation

2 In the second step, both domain experts and ontology engineers collaborate to define the knowledge of
 3 the layered ontology network.

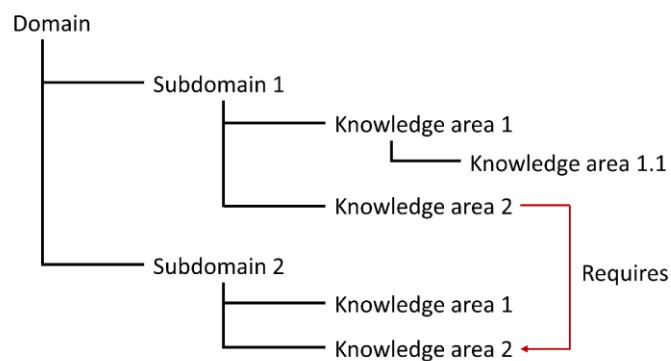
4 In previous reusable and usable ontology design methodologies, the knowledge of the layered ontology
 5 network is defined at a conceptual level. In addition, the knowledge is divided into different abstraction
 6 levels and knowledge pieces. This knowledge decomposition enables (1) the separation of abstract
 7 knowledge that is likely to be reused in most of applications from the specific knowledge and (2) the
 8 classification of the defined knowledge pieces into different abstraction levels (Spyns et al., 2008).

9 In previous methodologies, the knowledge of the layered ontology network is defined from scratch.
 10 However, MODDALS classifies the ontology domain knowledge based on a domain analysis of existing
 11 ontologies. Hence, the layered ontology network includes the knowledge represented by existing
 12 ontologies. In this step, the knowledge from existing ontologies is abstracted, divided and organised into
 13 a knowledge hierarchy that classifies it into different abstraction levels.

14 The knowledge hierarchy proposed by MODDALS includes three main elements (Fig. 6):

- 16 • *Domains*: the domains represented by the layered ontology network are located in the first level
 17 of the hierarchy.
- 18 • *Subdomains*: extensive domains are divided into subdomains that cover the knowledge of an
 19 important part of the domain. Hence, subdomains are located in the second level of the knowledge
 20 hierarchy.
- 21 • *Knowledge areas (KAs)*: in the third level of the knowledge hierarchy, consider a KA as a
 22 potential module of the layered ontology network that encompasses the knowledge of a specific
 23 topic of a subdomain. The KAs are the knowledge pieces that are classified into different layers.
 24 Each KA can be divided into “child” sub-KAs that represent more specific knowledge. Therefore,
 25 we can say that a sub-KA extends the knowledge of a specific KA. In addition, some KAs may
 26 represent specific knowledge by combining the knowledge from other KAs. In these cases, the
 27 former KAs require the knowledge from the latter. These relations are also reflected in the
 28 knowledge hierarchy.

29



30

31

32

Fig. 6: Domain knowledge hierarchy example

33 Bearing in mind this hierarchical structure, the KAs of the upper levels include abstract domain
 34 knowledge, while the KAs of low levels include more specific domain knowledge. Hence, the knowledge

1 hierarchy enables to abstract and divide the knowledge from existing ontologies, so that the defined KAs
2 can be classified in the next steps into the layers defined in Step 1.

3 Before explaining the knowledge hierarchy, it is important to distinguish the knowledge it includes
4 from the knowledge of existing ontologies. The knowledge hierarchy includes the knowledge of existing
5 ontologies at the conceptual level, as a set of concepts and relations. On the contrary, ontologies include
6 this knowledge implemented through classes, properties and axioms used to represent the concepts and
7 relations.

8 To define the hierarchy, the domain experts and ontology engineers collaborate to perform a manual
9 analysis of the elements of existing ontologies in an ontology editor to identify the domains they represent
10 and to divide them into KAs.

11 This step includes three activities that are conducted sequentially.

12
13 **1. Domain/subdomain definition:** in this activity, domain experts and ontology engineers analyse the
14 knowledge represented by exiting ontologies to identify the domains they represent. The top-level
15 concepts of each domain are also defined by domain experts. If the domains are too extensive, they
16 are divided into subdomains according to domain experts' criteria.

17 **2. Knowledge area definition:** in this activity, ontology engineers (in collaboration with domain
18 experts) analyse existing ontologies to divide the knowledge of the defined subdomains into KAs.

19 Ontology partitioning and module extraction algorithms/tools (D'Aquin, 2012; Grau, Horrocks,
20 Kazakov, & Sattler, 2008; Romero, Kaminski, Grau, & Horrocks, 2015) are well-known methods to
21 extract semi-automatically and divide knowledge from ontologies (D'Aquin, 2012). However,
22 existing ontologies are developed by different engineers and with different objectives, so they are
23 heterogeneous. Thus, to the best of our knowledge, the application of existing ontology partition and
24 module extraction algorithms/tools in different ontologies would lead to different ontology module
25 classifications. The same knowledge extracted from different ontologies may be included into
26 different modules and linked with different knowledge. These issues would lead to an inconsistent
27 knowledge hierarchy. Hence, a more abstract method to define KAs is required in MODDALS. To
28 avoid these issues, the Competency Questions (CQs; Grüninger & Fox, 1995) answered by existing
29 ontologies can be taken as reference to divide the knowledge they represent into KAs. The CQs
30 correspond to the queries that the ontologies must answer to the applications that query the
31 represented knowledge. Thus, they are a well-stablished method to define the ontology requirements
32 and the knowledge they must represent at a conceptual level (Suárez-Figueroa, 2010).

33 To answer each CQ, the ontology must include the necessary ontology elements (classes, properties
34 and axioms) that represent certain concepts and relations. Hence, CQs are a natural guide for splitting
35 ontologies into small knowledge fragments (Ruy, Guizzardi, Falbo, Reginato, & Santos, 2017). By
36 identifying the CQs each ontology answers, the concepts and relations needed to answer them can
37 also be identified and considered as a whole to define a KA. Hence, this method enables the
38 abstraction and division of knowledge from different ontologies regardless of their heterogeneous
39 knowledge representation.

40 The CQs defined to develop ontologies are not always available (Ruy et al., 2017). Therefore, in
41 MODDALS ontology engineers perform a manual analysis of ontology elements to identify the CQs
42 they answered by existing ontologies (it can be considered as a reverse engineering process) and
43 divide the knowledge into KAs. This strategy is also followed in when designing SPL taking as
44 reference existing applications (Breivold, Larsson, & Land, 2008; Harhurin & Hartmann, 2008), since
45 "*legacy systems rarely have an accurate functional specification*" (Harhurin & Hartmann, 2008). In
46 particular, the requirements and functionalities are extracted from the existing applications before
47 analysing their similarities and differences.

48 The knowledge area definition activity involves two sub-activities.

1 **2.1 Class hierarchy-based KA definition:** in ontologies, the classes are organised into class
 2 hierarchies. Class hierarchies classify the ontology classes into a hierarchy where classes that
 3 represent abstract concepts are at the top and the classes that represent specific concepts are
 4 at the bottom. In the hierarchy, a class can subsume classes that represent more specific
 5 concepts or be subsumed by other classes that represent more abstract concepts (Hebeler et
 6 al., 2011).

7 Some ontology class hierarchies are self-descriptive enough to answer a set of CQs. Hence,
 8 the class hierarchies of existing ontologies are analysed to identify the first CQs. For instance,
 9 a class hierarchy that contains the *Device* class with more specific devices (i.e., appliance,
 10 sensor) as subclasses can answer the following CQ: *what type of devices are there?* Hence,
 11 the *devices KA* corresponding to this CQ could be defined. This KA would encompass the
 12 *device* concept and all the concepts (i.e., appliance, sensor) represented by subclasses of the
 13 *Device* class. Considering this, the first KAs of the knowledge hierarchy are defined based
 14 on some class hierarchies of existing ontologies and each level of these class hierarchies can
 15 be considered as sub-KA of the previous level. These KAs are named as the subject of the
 16 CQ they answer. In the previous example, the subject of the CQ was *devices*, so the KA
 17 should be called *devices*.

18 However, the existing ontologies may represent the same concepts with different class
 19 hierarchy structures. Therefore, a common class hierarchy of these concepts must be defined
 20 before defining the KAs. In these cases, the class hierarchy that describes each concept with
 21 the highest granularity is selected among existing ontologies and is populated with classes
 22 from other ontologies according to the domain expert criteria.

23 **2.2 Ontology elements relation-based KA definition:** the rest of CQs are answered through the
 24 relations of a set of ontology elements. Hence, the classes of the existing ontologies and the
 25 relations between classes through properties (and the axioms applied on them) are analysed
 26 to identify the remaining CQs. All the concepts and relations represented by the ontology
 27 elements that answer each CQ can conform a KA. The CQs that cover similar topics are
 28 grouped by domain experts to create new KAs, which encompass all the knowledge required
 29 to answer these CQs. Each of these KAs is named by joining the key words of the CQs it
 30 encompasses. For example, let us consider that the analysed ontologies contain the *hasName*,
 31 *hasModel* and *hasSerialNumber* properties to describe certain features of *Devices* to answer
 32 the following CQs: *What is the name of a device?*, *What is the model of a device?* and *What*
 33 *is the serial number of a device?*. These CQs describe the information of the device related
 34 with the manufacturer, so they can be grouped into the *device manufacturer data KA*. This
 35 KA encompasses the concepts and relations that answer the aforementioned CQs.

36 By grouping CQs, some KAs may include unnecessary knowledge for certain applications.
 37 However, if we define one KA for each identified CQ, the knowledge hierarchy would
 38 contain an unmanageable number of KAs and thus the layered ontology network would
 39 contain an unmanageable number of modules (Ruy et al., 2017). We must assume that “*an*
 40 *ontology is never ready for use, but must always be adapted and refined to a knowledge base*
 41 *for the envisioned application*” (Morbach et al., 2009). Therefore, the CQs are grouped
 42 according to domain experts’ criteria and the desired KA classification granularity.

43
 44 **3. Knowledge hierarchy refinement:** in this activity, domain experts classify each KA into one
 45 domain/sub-domain and one level of the knowledge hierarchy, according to the knowledge that the
 46 KA represents or extends. In addition, they define the dependencies between KAs. If two KAs require
 47 the knowledge of each other, they are joined into a single one to avoid circularity and an inconsistent
 48 knowledge hierarchy.

1 Finally, domain experts provide a complete description of each KA (with the concepts and relations it
2 should include) and write the CQs it encompasses, to explain the knowledge included by the KA and
3 when it should be considered as represented. For instance, a description of the sample *device*
4 *manufacturer data* KA introduced in sub-activity 2.2 could be the following: “*this knowledge area*
5 *encompasses all the knowledge used to represent the device features related with the manufacturer (i.e.,*
6 *brand, model, serial number). It does not encompass device features related with operational aspects*
7 *(i.e., power, height)”*.

8
9 The outcome of this step is the knowledge hierarchy and the description of KA.

10 3.4. Step 3: Knowledge Classification

11 In the third step, ontology engineers classify the KAs defined in Step 2 into each abstraction layer.

12 A domain analysis of existing ontologies is performed by applying well-known SPL engineering
13 techniques to classify the knowledge, since it is one of the core requirements of MODDALS. We defined
14 this step based on the well-known domain analysis techniques and guidelines proposed by Pohl et al.
15 (2005) and Moon et al. (2005), which were adapted to be applied in the ontology engineering field.

16 Before conducting the domain analysis, domain experts analyse the defined KAs to identify the ones
17 that must be common due to their relevance to the domain because they represent abstract concepts and
18 relations. These KAs are directly included in the *common-domain layer* regardless of its presence in
19 existing ontologies, what has influence in the ontology knowledge classification. If the classification of
20 these KAs depended only on their presence in existing ontologies, they might be classified in low-level
21 layers although being relevant for the domain. Hence, as well as in the SPL design process, the domain
22 experts have influence in the knowledge classification, which is not 100% dependent on existing
23 applications (Pohl et al., 2005).

24 The rest of KAs are classified according to the domain analyses of existing ontologies. This step
25 includes five activities, which are conducted sequentially.

26
27 **1. Analysis of existing ontologies:** existing ontologies are analysed by ontology engineers to see
28 whether they represent the KAs defined in Step 2. It is worth mentioning that this analysis has a
29 different purpose and is more exhaustive than the one conducted in Step 2. In Step 2, the ontologies
30 are analysed to identify and divide the knowledge they represent into KAs. In this step the ontologies
31 are analysed to identify how many of them represent the defined KAs.

32 We consider that an ontology represents a KA if it includes the necessary elements
33 (classes/statements/axioms) to answer at least one of the CQs encompassed by the KA concerned. A
34 related point to consider is that if a “child” KA is represented by the ontology, the “parent” KA that
35 represents more abstract knowledge is considered represented. This rule avoids the placement of
36 abstract concepts in lower level layers than the specific concepts that extend the abstract concepts.

37 Most of the ontology analysis is performed manually by the ontology engineer by examining in the
38 ontology editor for the elements that represent the data encompassed by each KA. To identify faster
39 the ontology elements that represent the knowledge of the KA, the ontology engineer can use the
40 tools available in the editor (i.e., search engines) to find the key words of the KA and its
41 description/CQs in the ontology elements.

42 Apart from ontology engineers, domain experts also take part on this activity. They can assist
43 ontology engineers with additional explanations and clarifications about the defined KAs. This
44 collaboration helps ontology engineers to understand better the knowledge encompassed by a KA
45 when it is not clear whether the KA is represented by an ontology.

2. **Commonality and Variability Analysis (CVA):** the CVA is the main activity of the domain analysis. It is the process of identifying and classifying the software common and variant features (Pohl et al., 2005). In MODDALS, ontology engineers conduct a CVA of existing ontologies to determine whether the KAs of each subdomain are common to application types. There are two types of techniques to perform a CVA: the application requirements-matrix and the priority-based variability and checklist based variability analysis. The former classifies software features into common and variant depending on how many applications require them. The latter classify software features into common and variant depending on stakeholders’ priorities. In MODDALS, a CVA is applied to determine if the KAs are common to application types based on their presence or not in existing ontologies. These ontologies already include the knowledge defined by domain experts and the application stakeholders. The priority-based variability and checklist based variability analysis would involve defining a great part of the common and variant knowledge from scratch and doing meetings with stakeholders to establish their priorities. Hence, we selected the *application-requirements matrix* to apply it in MODDALS among existing CVA techniques.

To define this step, we took as reference the application-requirements matrix-based CVA conducted by Moon et al. (2005), since it explains how to the apply application-requirements matrix technique through an application example. Since the CVA is conducted to identify common and variant domain knowledge, we defined a new term for the matrix: the *application-knowledge matrix*. An example of the application-knowledge matrix template we propose in MODDALS is shown in Table 1. The left column contains the KAs of a specific subdomain (i.e., knowledge area 1, knowledge area 1.1). The top rows list different application types and the ontologies (i.e., ontology 1 (O1), ontology 2 (O2)) according to the application type they support. The matrix indicates if an ontology represents a KA (‘X’) or not (‘-’). With this information, the ontology engineer deduces which application types reuse each KA. We consider that an application type reuses a KA if the KA is represented by at least one ontology that provides support to the application type.

To determine whether a KA is common or variant, their *Commonality Ratio* (CV ratio) is taken as a reference (Moon et al., 2005). In this case, the CV ratio is the ratio of the number of application types that reuse a specific KA to the total number of application types. For instance, in Table 1 the *knowledge area 1* is reused by all application types, so it has a CV ratio of 100%. To the best of our knowledge, there is no systematic method to determine the exact threshold value of the CV ratio to identify common and variant software features. The CVAs conducted in the SPL engineering field (Breivold et al., 2008; Moon et al., 2005; Nestor, O’Malley, Quigley, Sikora, & Thiel, 2007) consider as common features the ones that are present in most of applications.

Ontologies	Application type 1			Application type 2		Application type 3	Application type 4		Commonality Ratio
	O1	O2	O3	O4	O5	O6	O7	O8	
Knowledge area 1	X	-	-	X	-	X	X	X	100%
Knowledge area 1.1	X	X	-	X	X	X	X	X	100%
Knowledge area 1.2	X	X	-	-	X	-	-	-	50%
Knowledge area 2	X	X	X	X	-	-	X	-	75%
Knowledge area 3	-	-	-	X	X	-	-	-	25%
Knowledge area 4	-	-	-	X	-	-	-	-	25%

Table 1: Example of an application-knowledge matrix

Thus, in MODDALS the ontology engineer determines CV ratio threshold depending on the number of the application types included in the domain analysis. In the example, there are four application types, so we can consider 75% as threshold value to distinguish between common and variable KAs. The common KAs are the ones that equal or exceed the threshold CV, while the rest of KAs are considered variant.

3. Knowledge area layer assignment: ontology engineers place the KAs in different layers according to the CVA results. Common KAs are placed in the *common-domain layer*. Variant KAs reused by more than one application type are assigned to the *variant-domain layer*.

Variant KAs reused only by one application type are placed in the *domain-task layer*. In addition, the KAs of this layer are classified according to the application type that reuse it.

4. CVA at the application type level: if the *domain-task layer* includes two sublayers to represent the knowledge of general and specific application types, another CVA at the application type level is required. Ontology engineers conduct this CVA to determine if KAs of this layer are relevant to the general application type or only to the specific application type. The KAs reused by more than one specific application types are likely to be reused by more future specific application types. Thus, these KAs are considered relevant to the general application type and they are placed in the *general application type sublayer*. The KAs reused only by a specific application type are assigned to the *specific application type sublayer*. The CVA at the application type level is applied to check if KAs are reused by one or more specific application types, so the CV ratio is not taken as a reference. According to the results of the example CVA (Table 1), *knowledge area 3* and *knowledge area 4* are only reused by *application type 2*. If we consider that this application type encompasses more specific application types (*application type 2.1*, *application type 2.2* and *application type 2.3*) a CVA at the application type level is conducted (Table 2). According to the CVA results, *knowledge area 3* is placed in in the *general application type sublayer* and *knowledge area 4* is placed in the *specific application type sublayer*.

	Application type 2		
	Application type 2.1	Application type 2.2	Application type 2.3
Knowledge area 3	X	X	-
Knowledge area 4	-	-	X

Table 2: CVA at application type level

The outcome of the *knowledge classification* step is a list of the KAs of each layer/sublayer.

3.5. Step 4: Layer Knowledge Structuring

The last step is to define how the knowledge of each layer defined in step 1 is structured. This step is conducted by ontology engineers and takes as input the knowledge hierarchy defined in Step 2 (see Section 3.3) and the KA classification obtained in Step 3 (see Section 3.4).

The ontologies that follow the structure designed with MODDALS will correspond to layered ontology networks reused by different applications. Hence, the knowledge of the layers must be structured to facilitate ontology reuse, as well as the inclusion of new knowledge to support new applications. To meet these requirements, previous reusable and usable ontology approaches (Morbach et al., 2009; Thakker et al., 2011) structure the knowledge of each layer into ontology modules and define the high-level relations between them when designing the layered ontology structure. In addition, they apply the main principles of ontology modularisation: loosely coupling and self-containment. These principles establish that an

1 ontology module must depend as little as possible on other modules to ease their understanding, reuse
2 and maintenance (D'Aquin, 2012; Stuckenschmidt & Klein, 2003).

3 Considering these principles, this step includes two activities, which are performed by the ontology
4 engineers and conducted sequentially.

5
6 **1. Ontology modularisation:** the ontology engineers classify the KAs of the layered ontology network
7 into different modules, which are defined in the following cases:

- 8 • An ontology module is defined to include the top-level concepts of each domain and placed in
9 the *common-domain layer*. The ontology module takes its name from the domain or the top-level
10 concept (if the module includes only one concept). In this way, we abstract the knowledge that is
11 extended by the rest of ontology modules.
- 12 • An ontology module is defined for each KA (the module encompasses the knowledge of the KA),
13 and placed in one ontology layer/sublayer according to the domain analysis results. The ontology
14 module takes its name from the name of the KA. There are two special cases where further
15 classification is required. (1) The KAs of the *common-domain layer* are likely to be reused in
16 most ontologies derived from the layered ontology network. Hence, the KAs of each subdomain
17 that belong to the *common-domain layer* are grouped into a single module that represents the
18 subdomain common domain knowledge. (2) The ontology modules of the *domain-task layer* are
19 classified according to the application type where the KA is reused.

20 **2. Inclusion hierarchy definition:** the ontology engineers organise previously defined ontology
21 modules into an inclusion hierarchy that establishes the high-level relations between the ontology
22 modules. Each ontology module must include only the modules whose knowledge extends or requires.
23 These relations define how the modules will be linked during the implementation of the layered
24 ontology network. The relations between modules are defined taking as reference the relations
25 between KAs in the knowledge hierarchy defined in Step 2. Hence, only the ontology modules that
26 represent closely related topics are related and their relations are limited. This ontology module
27 independency will enable an easier reuse of individual modules when constructing application
28 ontologies and the customisation of particular modules without affecting other modules when reusing
29 and extending the layered ontology network (Morbach et al., 2007).

30 As summary and example of this step, Fig. 7 shows how the KA classification is mapped into an
31 ontology module hierarchy.

32

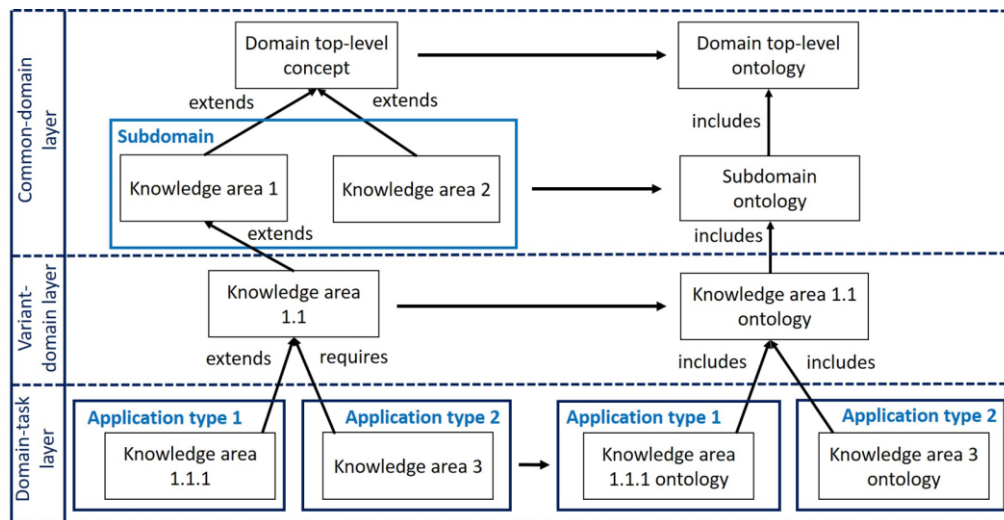


Fig. 7: Ontology modularisation and inclusion hierarchy definition

1
2
3
4
5
6
7
8

The outcome of this step is the informal model that contains the layered structure of the layered ontology network. This model corresponds to a schema that includes the ontology modules of each layer and the high-level relations between modules. The informal model also includes the descriptions of the knowledge of each module at a conceptual level. These descriptions are taken from the descriptions of KAs made in Step 2.

9 4. Application of MODDALS in the Energy Domain

10 This section illustrates how the MODDALS methodology was applied in a real use case for designing
 11 the layered ontology structure of DABGEO ontology (Domain Analysis-Based Global Energy Ontology)¹.
 12 DABGEO is a reusable and usable global ontology for the energy domain developed to be reused by
 13 energy management applications. The development of a global ontology is a key challenge to be
 14 addressed in the energy domain (Cuenca, Larrinaga, Eciolaza, & Curry, 2019). The DABGEO ontology
 15 is quite extensive (it includes 97 ontology modules), so the next subsections explain through examples
 16 how a certain parts of the layered ontology structure was designed. The same process was followed to
 17 design the rest of the layered ontology structure.

18 4.1. Preliminary Step: Analysis and Classification of Existing Energy Ontologies

19 The authors conducted a state of the art of existing energy ontologies and the application types to which
 20 they provide support (Cuenca et al., 2019).

21 According to this study, ontology-based energy management applications were classified into different
 22 types according to the Smart Grid scenario where they are deployed such as Smart Homes or organisations.
 23 The energy management application types for which energy ontologies are already developed include (1)
 24 Smart Home energy management applications, (2) building/district/city energy management applications,
 25 (3) organisation energy management applications and (4) Smart Grid Demand Response energy
 26 management applications. We define these application types as *Smart Grid scenarios* (Cuenca et al.,

¹ <http://www.purl.org/dabgeo>

1 2019). Each Smart Grid scenario encompasses more specific application types. For example, Smart Home
 2 energy management applications encompass home energy assessment, home energy saving advice, and
 3 home appliances DR management applications.

4 Finally, the relevant and available energy ontologies were classified according to the Smart Grid
 5 scenarios and specific energy management application types to which they provide support. This
 6 classification is shown later in Section 4.4.

7 *4.2. Step 1: Definition of DABGEO Ontology Layers*

8 The layers proposed in Step 1 of the MODDALS methodology (see Section 3.1) was defined for
 9 DABGEO by the domain experts taking as reference the ontology classification obtained in the
 10 preliminary step.

11 Following this structure, DABGEO includes three layers. The *common-domain layer* represents the
 12 top-level knowledge of energy domains and the knowledge common to Smart Grid scenarios. Variant
 13 domain knowledge still common to more than one Smart Grid scenario is included in the *variant-domain*
 14 *layer*. The *domain-task layer* includes the knowledge reused in specific Smart Grid scenarios and is
 15 divided into two sublayers: the *Smart Grid scenario* and the *application type* sublayers. The former
 16 represents the knowledge relevant to a certain Smart Grid scenario and the later represents the knowledge
 17 reused only by certain energy management application types of a Smart Grid scenario. The domain
 18 experts named each sublayer to facilitate the distinction between both sublayers.

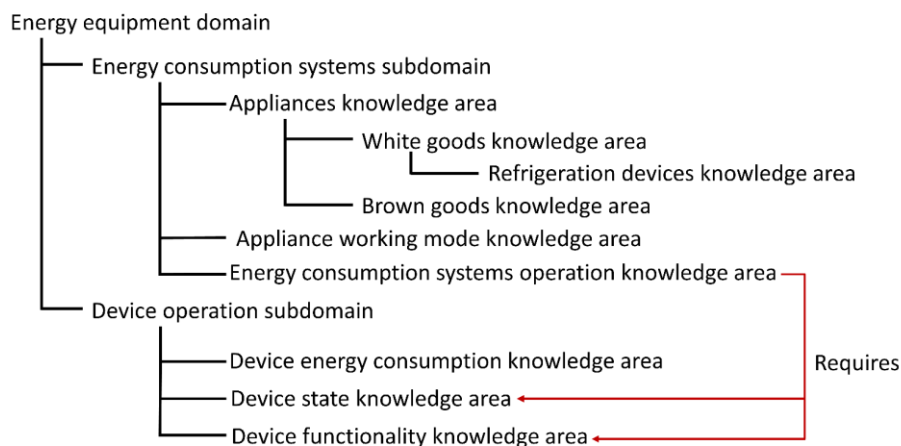
19 *4.3. Step 2: DABGEO Knowledge Hierarchy Definition*

20 In this step, the domain knowledge hierarchy of DABGEO was defined.

21 Fig. 8 shows part of the whole knowledge hierarchy of DABGEO. Since the DABGEO domain
 22 knowledge was classified based on a domain analysis of existing energy ontologies, the knowledge
 23 hierarchy includes the knowledge represented by existing energy ontologies. The domain experts and
 24 ontology engineers collaborated to perform a manual analysis of ontology elements in Protégé to identify
 25 the domains they represent and to divide them into KAs.

26 Below we describe how Step 2 activities were conducted to define the part of the knowledge hierarchy
 27 shown in Fig. 8.

28



29
 30

Fig. 8: Part of DABGEO knowledge hierarchy for the energy equipment domain

- 1 **1. Domain/subdomain definition:** in this activity, domain experts and ontology engineers analysed the
2 knowledge represented by existing energy ontologies to identify the domains they represent:
3 a. *Energy equipment domain:* the features and operation data about energy consumption
4 production and storage devices.
5 b. *Infrastructure domain:* data on structural features and environmental conditions of
6 infrastructures such as homes or buildings.
7 c. *Energy performance domain:* data on energy performance values and indicators such as
8 energy consumption or production.
9 d. *Energy external factors domain:* data on factors that may hinder the energy performance such
10 as weather or environmental conditions.
11 e. *Smart Grid stakeholders domain:* data on the actors that participate in the energy market such
12 as energy consumers and producers.

13 In addition, the root concepts of each domain were defined by domain experts. For instance, *device*
14 was defined as the root concept of the *energy equipment domain* because this concept is extended by
15 the rest of the data (device types, device operation data) included in the domain.

16 These domains were divided into subdomains by domain experts because they are extensive. For
17 instance, many concepts are needed to describe the whole *energy equipment domain*, since this
18 domain encompasses data about many device types and their operational aspects. Hence, this domain
19 was divided into the *energy consumption systems* and *device operation* subdomains, among others
20 (see Fig. 8). The former contains knowledge about energy consumption devices such as appliances
21 or heating systems. The latter represents functional features about devices such as device state or
22 device functionality.

- 23 **2. Knowledge area definition:** in this activity, the ontology engineers (in collaboration with domain
24 experts) analysed the existing energy ontologies to identify the CQs they answered. The CQs were
25 taken as reference to divide the knowledge of existing energy ontologies into KAs. In total, 10 energy
26 ontologies were analysed, including ThinkHome² and EnergyUse³. Below we explain how the sub-
27 activities of the *knowledge area definition and classification* activity were conducted to define some
28 sample KAs within the *energy consumption systems* and *device operation* subdomains.
29

30 **2.1 Class hierarchy-based KA definition:** firstly, the class hierarchies of the energy ontologies
31 were analysed by ontology engineers to identify the CQs. Regarding energy consumption
32 systems data, the energy ontologies represent the *Appliance* class and more specific
33 appliances as subclasses of this class. Therefore, one of the CQs answered by the class
34 hierarchies is *What type of appliances are there?* Hence, the *appliance KA* was defined,
35 which encompasses the *appliance* concept and the concepts represented by the subclasses of
36 the *Appliance* class. The ThinkHome ontology is the one that classifies appliances with more
37 granularity, so the class hierarchy of this ontology was taken as reference to define the
38 *appliance KA* and its sub-KAs. ThinkHome classifies the *Appliance* class into subclasses that
39 represent specific appliance types such as *Brown goods* and *White goods*, which, in turn,
40 encompass subclasses that represent specific white and brown good types. The class
41 hierarchy was populated with specific classes from other ontologies such as classes that
42 represent specific white goods (i.e., *Refrigeration devices*). Each of these classes were
43 defined as KAs that encompass the concepts of all their subclasses (see Fig. 8). In addition,
44 each KA of each class was defined as a sub-KA of the corresponding superclass.

² <https://www.auto.tuwien.ac.at/downloads/thinkhome/ontology/>

³ <http://socsem.open.ac.uk/ontologies/eu#>

1 Regarding the device operation data, the existing energy ontologies answer the following
 2 CQs: *What are the device functionality types, What are the device state types?*. Hence, the
 3 *device functionality* and *device state* KAs were defined.

4 **2.2 Ontology elements relation-based KA definition:** the remaining KAs were defined after
 5 identifying the CQs answered by a set of interrelated elements of existing energy ontologies.
 6 As an example, Fig. 9 and 10 show a set of ontology elements of ThinkHome and EnergyUse
 7 ontologies respectively within a Protégé screenshot. As marked (in red) in Fig. 9, the
 8 ThinkHome ontology includes the *consumesEnergy*, *actuallyConsumesEnergy* and
 9 *maxConsumesEnergy* properties. These properties describe the energy consumption, actual
 10 energy consumption and maximum energy consumption of a certain device respectively.
 11 Hence, the ThinkHome ontology answers the following CQs: *What is the energy
 12 consumption of a device?*, *How much energy is a device consuming?* and *What is the
 13 maximum energy consumption of a device?* On the other hand, as shown in Fig. 10, the
 14 EnergyUse ontology includes the *hasConsumption* property to answer the *What is the energy
 15 consumption of a device?* CQ. All these CQs describe energy consumption of devices, so
 16 they were grouped by the domain experts into the *device energy consumption* KA (which also
 17 includes CQs answered by other energy ontologies). This KA encompasses the knowledge
 18 that answers the aforementioned CQs. In the same way, the *energy consumption systems
 19 operation* and *appliance working mode* KAs were defined. These KAs encompass the
 20 knowledge about operational aspects of specific energy consumption systems and appliance
 21 working modes respectively.
 22

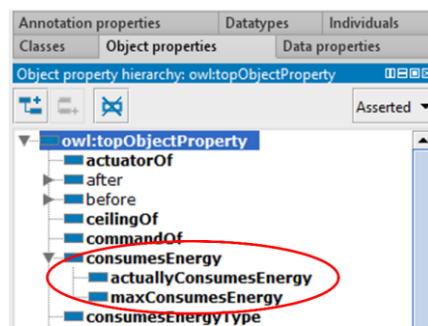


Fig. 9: Ontology elements of ThinkHome ontology

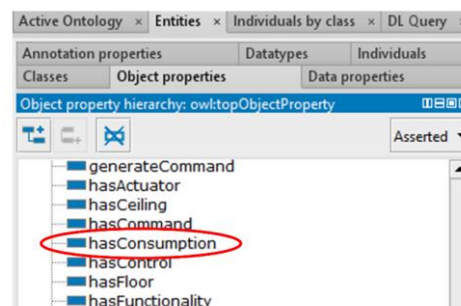


Fig. 10: Ontology elements of EnergyUse ontology

1 **3. Knowledge hierarchy refinement:** in this activity, the KAs were placed into a knowledge
 2 hierarchy level according to the knowledge they represent and extend, thus completing the
 3 knowledge hierarchy. Fig. 8 shows in which subdomain and hierarchy level was placed each
 4 KA introduced in previous examples. In addition, the KA dependencies were also defined. For
 5 instance, the *energy consumption systems operation KA* describes specific states and
 6 functionalities of energy consumption systems and encompasses CQs such as *What is the*
 7 *minimum number of states an air condition system has?* and *Do ventilating systems have any*
 8 *notification functionality?*. Therefore, this KA requires the knowledge of *device state* and
 9 *device functionality* KAs, which include knowledge about possible device states and
 10 functionalities respectively. Finally, the domain experts provided a complete description of
 11 each KA, with the knowledge and CQs it encompasses. As an example, the following are the
 12 descriptions of the brown goods and energy consumption systems operation KAs:

- 14 • **Brown goods:** “represents data about any small appliance such as coffee makers,
 15 office, entertainment equipment or multimedia devices. We consider that this
 16 knowledge area is represented by an ontology if any of these devices are represented
 17 or if there is a class that explicitly represents ‘brown goods’. This knowledge area
 18 encompasses the following sub-knowledge areas: IT equipment and entertainment
 19 equipment”.
- 20 • **Energy consumption systems operation:** “represents the states (i.e., on/off states)
 21 that any energy consumption system (i.e., heating ventilation and air conditioning
 22 (HVAC) systems, appliances or lighting systems) can have and functionalities (i.e.,
 23 state notification, command reception) that devices can perform”.

24 4.4. Step 3: DABGEO Knowledge Classification

25 A domain analysis of existing energy ontologies was conducted by the ontology engineers to classify
 26 the defined KAs into each layer.

27 Firstly, the domain experts included in the common-domain layers the KAs that represent relevant
 28 domain knowledge for the domain. Then, the following activities were conducted.

29
 30 **1. Analysis of existing ontologies:** existing energy ontologies were manually analysed with Protégé
 31 to determine if they represented the KAs of energy domains. Specifically, tools available in this editor
 32 were used to find the KA key words (extracted from the KA description provided by the domain
 33 expert) in the ontology elements. If the ontology contained necessary elements or statements to
 34 answer the CQs encompassed by the KA, the KA was considered as represented by the ontology. As
 35 an example, Fig. 11 shows a screenshot of a set of ThinkHome ontology classes that represent specific
 36 brown goods (i.e., alarm clock, entertainment equipment). Therefore, the ontology answers the CQ
 37 *what types of brown goods are there?*, which is encompassed by the *brown goods KA*. Taking this
 38 into account, we considered that the ThinkHome ontology represents this KA.

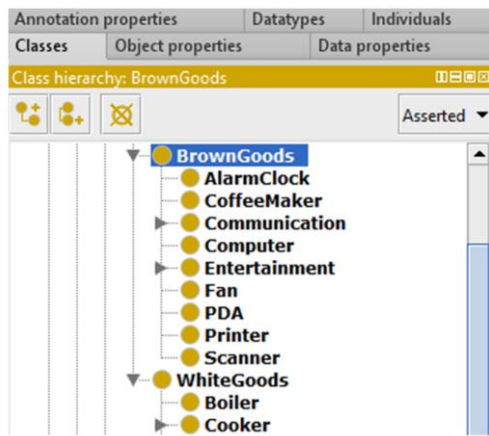


Fig. 11: Representation of the brown goods KA by ThinkHome ontology

The *brown goods KA* is an intuitive example that requires only the analysis of certain classes to determine whether the KA is represented. However, other KAs required a more exhaustive analysis, since they were represented by more specific classes and relations. Taking as an example the *device energy consumption KA* (described at the end of Section 4.3), only certain properties were applied to relate device operational aspects with specific energy consumption systems. Hence, a more exhaustive analysis of energy ontologies was performed to see whether they represent this KA.

2. Commonality and Variability Analysis: a CVA was conducted to identify common and variant energy KAs of each energy subdomain. An application-knowledge matrix of each energy subdomain was created to determine which Smart Grid scenarios reuse each subdomain KA, taking as reference the representation of these KAs by existing energy ontologies. As an example, Table 3 shows the application-knowledge matrix of some KAs of the *energy consumption systems* subdomain (the ones included on the knowledge hierarchy of Fig. 8). The left column includes the KAs, while the top row includes the Smart Grid scenarios and the ontologies that provide support to the applications deployed in these scenarios. To simplify the table, we omitted several ontologies. Since there are currently four Smart Grid scenarios for which ontologies were developed (according to the classification performed by Cuenca et al. (2019), 75% was used as the threshold value to classify the KAs as common or variant depending on their CV ratio.

Ontologies	Smart Grid scenarios								Commonality Ratio
	Smart Home energy management				Building/ district/ city energy management	Organisation energy management		Smart Grid Demand Response management	
Knowledge areas	ThinkHome ontology	EnergyUse ontology	SAREF4EE ontology	Mirabel ontology	SEMANCO ontology	DEFRAM project ontology	DERI Linked dataspace	ProSGV3 ontology	
Appliances	X	X	X	X	X	X	X	X	100%
Brown goods	X	X	-	-	-	-	X	X	75%
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 3: Application-knowledge matrix of the energy consumption systems subdomain

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

22

- 1 **3. Knowledge area layer assignment:** the KAs were classified into different layers according to the
 2 CVA results. For instance, the *appliances*, *brown goods* and *white goods* KAs were classified into
 3 the *common-domain layer*, since their CV ratio was equal of above 75%. The *refrigeration devices*
 4 *KA* was placed in the *variant-domain layer*, since it was common to more than one Smart Grid
 5 scenario although its CV ratio was below 75%.
- 6 **4. CVA at the application type level:** the KAs reused only one Smart Grid scenario were classified
 7 into the sublayers of the domain-task layer according to the CVA at the application type level.
 8 Following the sample CVA shown in Table 3, the *energy consumption systems operation* and the
 9 *appliance working mode* KAs were included in this domain analysis, since they were only represented
 10 by ontologies from Smart Home energy management applications. This low representation is because
 11 these KAs encompass the knowledge that answers very specific CQs that only ontologies reused in
 12 Smart Home energy management applications must answer. The domain analysis at the application
 13 type level for these KAs is shown in Table 4. The *energy consumption systems operation KA* was
 14 reused by more than one Smart Home energy management application type (*home energy assessment*
 15 and *home energy saving advice* applications), so it was placed in the *Smart Grid scenario sublayer*.
 16 The *appliance working mode KA* was reused only by one Smart Home energy management
 17 application type (*home appliances DR management*), so it was placed in the *application type sublayer*.
 18

		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 4: CVA at application level of energy consumption systems subdomain

19
20

21 **4.5. Step 4: Structuring of DABGEO Layer Knowledge**

22 Finally, the knowledge of each layer was structured into ontology modules by the ontology engineers,
 23 thus completing the design of DABGEO layered ontology structure.

24 Fig. 12 shows the informal model of part of DABGEO structure corresponding to the *energy*
 25 *consumption systems* subdomain. Below we detail how the activities of this step were carried out, taking
 26 as an example this subdomain.

27

28 **1. Ontology modularisation:** in Step 2, *device* was defined as the top-level concept of the *energy*
 29 *equipment domain* and, by extension of the *energy consumption systems subdomain* (see Section 4.3).
 30 Hence, the *Device ontology module* was defined, which represents the *Device* top-level concept and
 31 device main properties, i.e., device name. In addition, all the common KAs (i.e. appliances, white
 32 goods KAs) of this subdomain were grouped into the *energy consumption systems ontology*, which
 33 includes all the knowledge they encompass. Both ontology modules are placed in the *common-*
 34 *domain layer*.

35 Then, one ontology module was defined for each variant KA (i.e., *refrigeration devices ontology*),
 36 and these modules were classified into lower-level layers according to the domain analysis results.
 37 Within the *Smart Grid scenario* and *application type* sublayers, the ontology modules were classified

1 depending on the Smart Grid scenario or the specific energy management application type where the
 2 KAs they represent are reused.
 3 **2. Inclusion hierarchy definition:** the defined ontology modules were organised into an inclusion
 4 hierarchy that establishes the high-level relations between the ontology modules. The inclusion
 5 hierarchy was defined based on the knowledge that the ontology modules extend or require (taking
 6 as reference the knowledge hierarchy defined in Step 2). For instance, the *Device ontology* is included
 7 by the *energy consumption systems ontology*, which in turn is included by a set of ontology modules
 8 from lower-level layers.
 9

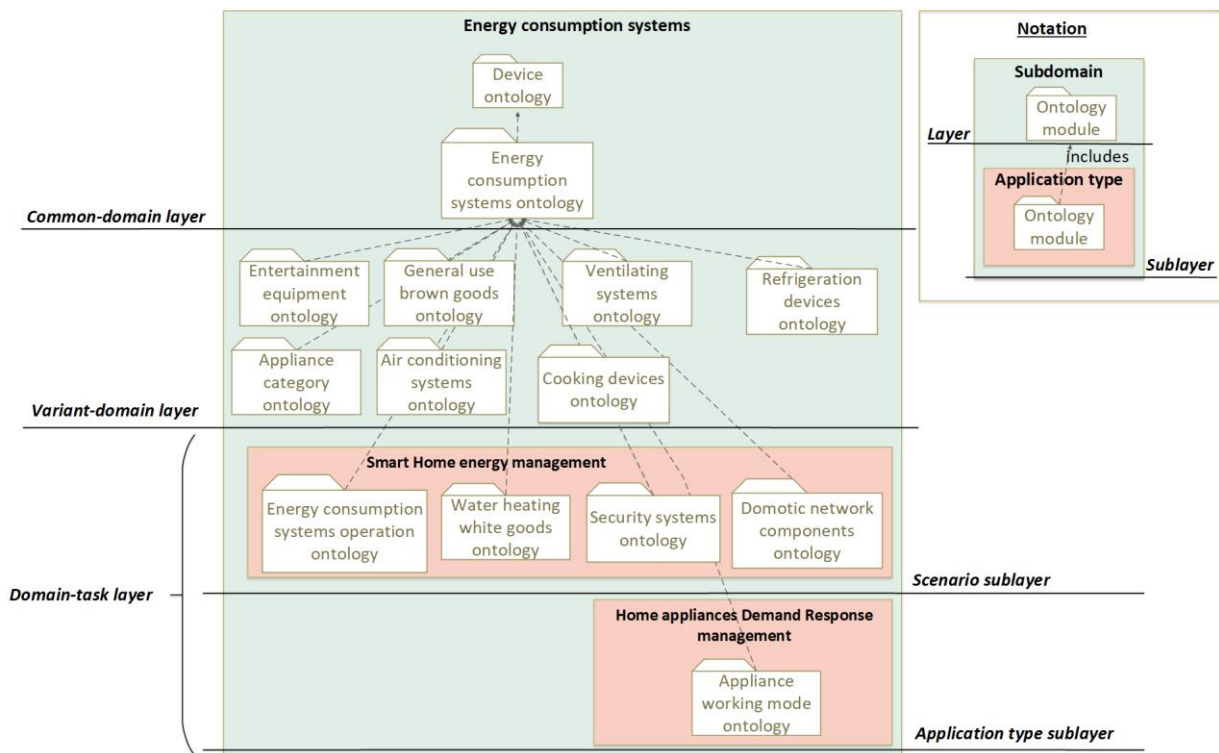


Fig. 12: Informal model of DABGEO structure concerning the energy consumption systems subdomain

5. Evaluation

13 As stated by De Hoog (1998), “it is extremely difficult to judge the value of a methodology in an
 14 objective way”. On the one hand, it is unlikely that anyone will be willing to pay twice for building or
 15 designing the same extended ontology using different approaches. On the other hand, the application of
 16 a methodology is a complex process where too many conditions cannot be controlled. Hence, the
 17 evaluation of previous ontology development and design methodologies consisted on showing the
 18 experiences of applying the methodology in one or more use cases (Khan & Keet, 2015; Kotis & Vouros,
 19 2006; Suárez-Figueroa et al., 2015). Considering this, we report in this section how we performed a first
 20 evaluation of the MODDALS methodology.

21 As stated in Section 1.1, the main requirement of MODDALS is to provide techniques to enable the
 22 classification of the domain knowledge taking as reference existing ontologies. Hence, the evaluation has
 23 focused on determining if MODDALS enables this classification. To demonstrate this aspect, we checked

1 whether MODDALS steps can be correctly followed by different domain experts and ontology engineers.
2 We consider that MODDALS steps can be followed correctly if different domain experts and ontology
3 engineers are able to obtain similar knowledge classifications performing a domain analysis of existing
4 ontologies.

5 To evaluate this aspect, MODDALS was applied by different energy domain experts and ontology
6 engineers to design a part of the layered structure of DABGEO. A group of domain experts and ontology
7 engineers conducted Steps 1 and 2, while the ontology engineers (eight in total) conducted Steps 3 and 4
8 with the collaboration of the experts. Each ontology engineer performed Steps 3 and 4 individually in a
9 blind process. However, they could contact the domain experts for any clarification or additional
10 explanation about the defined KAs to decide in which layer to place certain KAs. The knowledge
11 classifications obtained by each engineer are analysed to check if they are similar in Section 5.1.

12 In addition, to get the experiences of the domain experts and ontology engineers on applying
13 MODDALS, we performed a survey, which is a well-known method for evaluating methodologies (Palvia
14 & Nosek, 1990; Suárez-Figueroa et al., 2015). The survey includes a questionnaire that the participants
15 in the MODDALS evaluation answered to (1) identify MODDALS main benefits and drawbacks, (2)
16 identify future lines of research to improve the methodology and (3) determine whether it is ready to be
17 applied in other domains apart from the Energy. In Section 5.2, we show the responses to the
18 questionnaire.

19 5.1. MODDALS Application Results

20 In this section, we first show the energy knowledge classification obtained by different ontology
21 engineers after applying MODDALS to design part of DABGEO layered structure. To compare the
22 knowledge classifications and analyse whether they are similar, we analysed the number of modules
23 defined by each engineer in each layer. However, although the number of modules is the same, they may
24 contain different knowledge. Hence, the *degree of consensus* with which the ontology engineers classified
25 the KAs into different layers was also analysed. The degree of consensus of a KA is the percentage of
26 ontology engineers that classified the KA into the same layer.

27 Fig. 13 shows how many modules were defined by each engineer in each layer of the designed energy
28 ontology. Fig. 13 also shows the number of modules of the *domain-task layer* that were classified into
29 each energy management application type. It is worth mentioning that the *domain-task layer* did not
30 include any sublayer, since the designed ontology part was only limited to support three application types:
31 home energy saving advice, home appliances DR management and Smart Grid DR management
32 applications.

33 In general, the number of modules defined by each ontology engineer was similar in all layers. This
34 similarity is due to the high degree of consensus with which the ontology engineers classified the KAs
35 into different layers. Within the conducted evaluation, the average degree of consensus of all the KAs
36 classified by the ontology engineers was 76%. It is worth mentioning that from the sixth ontology
37 engineer that applied MODDALS onwards, the average degree of consensus remained stable in 76%.
38 Therefore, the MODDALS evaluation participants obtained similar ontology designs.
39

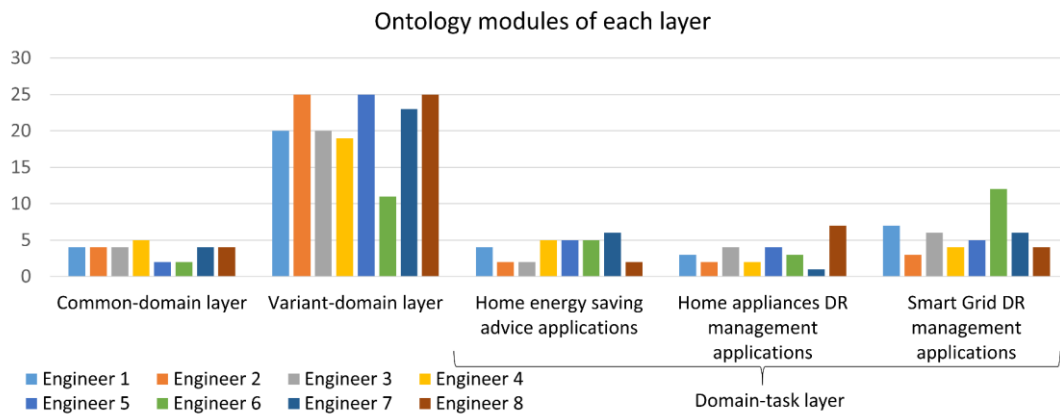


Fig. 13: Ontology modules of each layer

Most of the KAs (specifically 80%) whose degree of consensus was above the average (76%) were classified into the *common-domain* and *variant-domain* layers. As an example, some of these KAs, as well as their degree of consensus and the layer/application type were these KAs were placed, are shown in Fig. 14. Therefore, we can conclude that there was a high consensus when separating the common domain knowledge from the variant knowledge reused by specific application types.

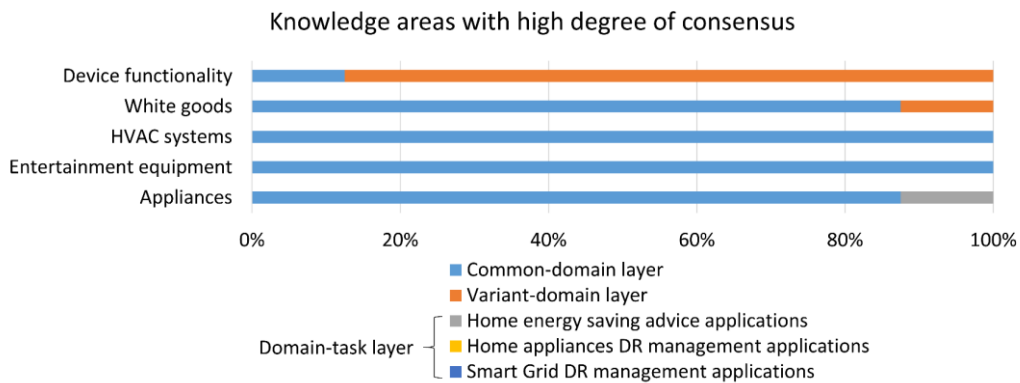


Fig. 14: KAs with high degree of consensus

Although ontology engineers could contact the domain experts for any clarification about the knowledge the KAs encompass, each ontology engineer had their own interpretation about the knowledge represented by existing ontologies. Thus, the degree of consensus of some KAs was lower (some examples are shown Fig. 15). This aspect constitutes one of the drawbacks of MODDALS, as we discuss later in Section 5.2.

A significant part (62%) of the KASs with low degree of consensus are child KAs of KAs whose degree of consensus is above the average (76%). Therefore, most of the differences in the classification of knowledge occurred in KAs that represent very specific knowledge, without affecting the rest of the classification.

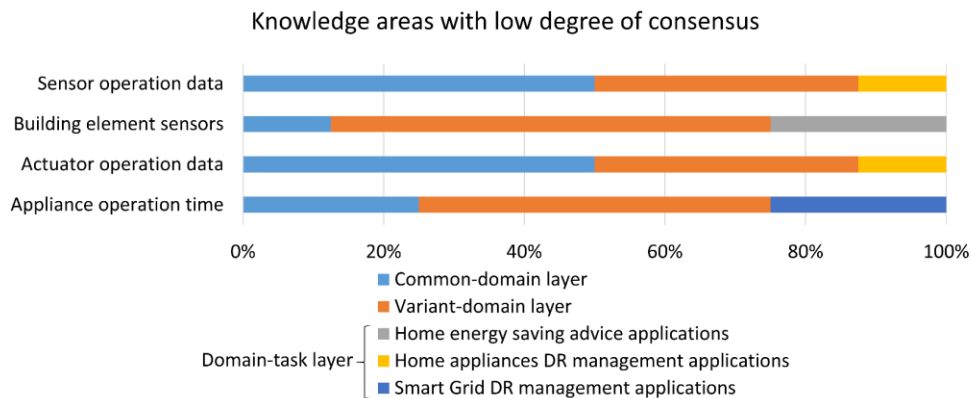


Fig. 15: KAs with low degree of consensus

Considering these results, domain experts and ontology engineers could follow MODDALS steps to obtain similar knowledge classifications. This classification was performed based on a domain analysis of existing ontologies, which complemented domain experts and ontology engineers' experience. Therefore, we can state that MODDALS can be applied by different domain experts and ontology engineers with similar knowledge classification results, enabling to classify the domain knowledge by taking as reference existing ontologies.

5.2. MODDALS Feedback

This section explains the responses of MODDALS evaluation participants to the questionnaire we provided for feedback on the methodology. The questionnaire included the following questions:

- What are the positive aspects of MODDALS?
- What are the disadvantages of MODDALS?

So far, we received 8 responses from participants involved in MODDALS evaluation. According to the survey respondents, the main benefits of MODDALS are the following:

1. Due to the domain analysis of existing ontologies, MODDALS provides a detailed classification of the knowledge reused by specific application types, while keeping separate the knowledge relevant to many applications. Some of the comments of survey respondents about this benefit were: "Common-domain layer starts with very general ideas and then it goes to more specific concepts in the next layers"; "It gives clear steps for determining which knowledge areas are common to existent ontologies and which knowledge areas are specific to certain ontologies/applications"; "ideology of multiple layers" [sic], "designed ontologies are likely to provide a balance between reusability and usability"; "It is very useful to compare different ontologies and identify which aspects are common on them".
2. MODDALS is easy to follow and provides clear and mechanical steps. Some of the comments of survey respondents about this benefit were: "It gives clear steps (mostly mechanical)"; "It is a simple process", "easy approach".
3. MODDALS provides a method to improve the reuse of already developed knowledge to enable the development of interoperable ontologies. *It seems a good method for refactoring already*

1 *available ontologies without discarding what it has been applied in the domain and enhancing*
2 *interoperability”.*

3 On the other hand, the following are the main disadvantages of MODDALS according to the survey
4 respondents:

- 5 1. Although it prevents domain experts and ontology engineers from designing the ontology
6 structure from scratch, MODDALS still requires a significant manual ontology analysis effort to
7 check if each KA is represented by existing ontologies. Some of the comments of survey
8 respondents about this disadvantage were: *“It requires much time to perform the domain analysis*
9 *of existing ontologies”* [sic]; *“identifying if the knowledge area is represented in the ontology is*
10 *not always straight forward for the ontology engineer”*; *“time-consuming process”*.
- 11 2. The classification of some KAs was mainly subject to ontology engineers’ interpretation of the
12 KA description provided by domain experts and the analysed knowledge from existing ontologies.
13 On the one hand, some of the KA descriptions were open to multiple interpretations. In addition,
14 a manual analysis of ontology engineers may not be sufficient to detect whether certain KAs are
15 represented, since part of the ontology knowledge may be implicit. Therefore, part of the domain
16 knowledge classification is quite subjective, which may influence the final design of the layered
17 ontology network. Some of the comments of survey respondents about this disadvantage were:
18 *“it depends on how well the knowledge area is described by the domain expert and how well*
19 *documented is the ontology”*, *“the role of the domain expert is catalytic”*; *“step three of the*
20 *methodology might create a bit of ambiguity”*; *“implied relationships might exist in an ontology,*
21 *and the end result might not have taken this into account”*; *“analysing the ontologies can be*
22 *subjective if the sub models are not well defined”*. This disadvantage is clearly reflected in the
23 results shown in Fig. 15 (Section 5.1). The average of the degree of consensus when classifying
24 KAs was of 76%. However, there were still KAs classified into different layers by different
25 ontology engineers, although they were defined by the same group of experts.
- 26 3. Although it enables the design of maintainable ontology structures, MODDALS does not provide
27 guidelines to extend the structure of layered ontology network and reclassify the knowledge when
28 new ontologies and applications arise. Some of the comments of survey respondents about this
29 disadvantage were: *“MODDALS guidelines are limited to design the first version of the ontology”*.

30 **6. Conclusions and Future Work**

31 In this paper, have we presented the MODDALS methodology. It guides domain experts and ontology
32 engineers to design the layered structure of reusable and usable ontology networks. The output of this
33 process is an informal model with the layers of the ontology network and the knowledge they include at
34 a conceptual level.

35 MODDALS is the result of combining the best practices of the ontology engineering and SPL
36 engineering fields. MODDALS adopts the main activities and ontology design principles applied by
37 previous reusable and usable methodologies to define the layered ontology structure. In contrast to these
38 methodologies, SPL engineering techniques are applied to classify the common and variant domain
39 knowledge into defined layers according to a domain analysis of existing ontologies. This approach
40 complements domain experts’ and ontology engineers’ expertise and prevents them from classifying the
41 domain knowledge from scratch, facilitating the design of the layered ontology structure.

1 MODDALS was applied by domain experts and ontology engineers to design the layered structure of
2 DABGEO, a reusable and usable global ontology for the energy domain. In that way, we illustrated how
3 this methodology is applied in a real use case.

4 MODDALS was evaluated to determine whether it enables to classify the domain knowledge by taking
5 as reference existing ontologies. Domain experts and different ontology engineers designed part of
6 DABGEO layered ontology structure by applying MODDALS. They were able to follow MODDALS
7 steps to obtain similar ontology designs by performing a domain analysis of existing ontologies (the
8 degree of consensus when classifying the domain knowledge was 76%). Hence, we can state that
9 MODDALS enables to classify the domain knowledge by taking as reference existing ontologies.

10 According to MODDALS evaluation participants, its main advantages are: (1) it provides a detailed
11 domain knowledge classification; (2) it is easy to follow and (3) improves the reuse of existing knowledge
12 to develop interoperable ontologies. By contrast, the main disadvantages of the methodology are: (1) the
13 knowledge classification step is time consuming due to the manual ontology analysis effort required and
14 (2) part of the knowledge classification is mainly subject to the subjective criteria of ontology engineers.
15 Hence, MODDALS is still a first step towards a widely accepted methodology to design layered ontology
16 structures for reusable and usable ontologies.

17 Considering MODDALS evaluation results, our current work is focused on automating the knowledge
18 classification step. In particular, we are exploring the possibility of integrating tools that semi-
19 automatically check whether certain ontologies answer a set of CQs. These tools would save manual
20 analysis effort of existing ontologies. The short-term term work will consist on (1) extending the
21 methodology to include guidelines to maintain the layered ontology structure and (2) applying
22 MODDALS in more domains to obtain more feedback and improve the methodology in future versions.
23 Finally, the evaluation of MODDALS was limited to determine whether it enables to classify the domain
24 knowledge by taking as reference existing ontologies. We consider that this approach will require less
25 time and effort than designing the layered ontology structure from scratch, as previous reusable and usable
26 ontology design methodologies do. To demonstrate that MODDALS reduces the effort of designing the
27 layered ontology structure, the time required to apply MODDALS should have been compared with the
28 time required by applying previous reusable and usable ontology design methodologies. This evaluation
29 corresponds to the mid-term future work, once automated the knowledge classification step.

30 Acknowledgements

31 This work has been developed by the Software and Systems Engineering group of MGEP and
32 supported by the Department of Education, Universities and Research of the Basque Government under
33 the projects Ikerketa Taldeak (Grupo de Ingeniería de Software y Sistemas, no. IT1326-19). We would
34 also like to thank all the colleges from Mondragon University and Insight Centre of Data Analytics who
35 applied MODDALS and who responded to the questionnaire we provided them to evaluate the
36 methodology.

37 References

- 38 Apel, S., Batory, D., Kästner, C., & Saake, G. (2016). *Feature-oriented software product lines*. Springer.
39 Breivold, H. P., Larsson, S., & Land, R. (2008). Migrating industrial systems towards software product lines: Experiences and
40 observations through case studies. *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro*
41 *Conference* (pp. 232–239). IEEE.

- 1 Chen, H., Perich, F., Finin, T., & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on* (pp. 258–267). IEEE.
- 2
3
- 4 Chen, L., Ali Babar, M., & Ali, N. (2009). Variability management in software product lines: a systematic review. *Proceedings of the 13th International Software Product Line Conference* (pp. 81–90). Carnegie Mellon University.
- 5
- 6 Choi, N., Song, I.-Y., & Han, H. (2006). A survey on ontology mapping. *ACM Sigmod Record*, 35(3), 34–41.
- 7
- 8 Cuenca, J., Larrinaga, F., Eciolaza, L., & Curry, E. (2019). Towards Cognitive Cities in the Energy Domain. *Designing Cognitive Cities* (pp. 155–183). Springer.
- 9
- 10 D’Aquin, M. (2012). Modularizing ontologies. *Ontology Engineering in a Networked World* (pp. 213–233). Springer.
- 11
- 12 De Hoog, R. (1998). Methodologies for building knowledge based systems: achievements and prospects. *The Handbook of Applied Expert Systems*.
- 13
- 14 Falbo, R. A. (2014). SABiO: Systematic Approach for Building Ontologies. *ONTO. COM/ODISE@ FOIS*.
- 15
- 16 Fantechi, A., Gnesi, S., John, I., Lami, G., & Dörr, J. (2003). Elicitation of use cases for product lines. *International Workshop on Software Product-Family Engineering* (pp. 152–167). Springer.
- 17
- 18 Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering.
- 19
- 20 Gómez-Pérez, A., Fernández-López, M., & Corcho, Ó. (2006). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media.
- 21
- 22 Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2008). Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31, 273–318.
- 23
- 24 Gruber, T. (2009). *Ontology*. Springer.
- 25
- 26 Grüninger, M., & Fox, M. S. (1995). Methodology for the design and evaluation of ontologies.
- 27
- 28 Guarino, N. (1997). Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology* (pp. 139–170). Springer.
- 29
- 30 Harhurin, A., & Hartmann, J. (2008). Service-oriented commonality analysis across existing systems. *2008 12th International Software Product Line Conference* (pp. 255–264). IEEE.
- 31
- 32 Hebel, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2011). *Semantic web programming*. John Wiley & Sons.
- 33
- 34 Jarrar, M., & Meersman, R. (2008). Ontology engineering—the DOGMA approach. *Advances in Web Semantics I* (pp. 7–34). Springer.
- 35
- 36 Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). *Feature-oriented domain analysis (FODA) feasibility study*. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.
- 37
- 38 Khan, Z. C., & Keet, C. M. (2015). An empirically-based framework for ontology modularisation. *Applied Ontology*, 10(3-4), 171–195.
- 39
- 40 Kotis, K., & Vouros, G. A. (2006). Human-centered ontology engineering: The HCOME methodology. *Knowledge and Information Systems*, 10(1), 109–131.
- 41
- 42 Maree, M., & Belkhatir, M. (2015). Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies. *Knowledge-Based Systems*, 73, 199–211.
- 43
- 44 Moon, M., Yeom, K., & Chae, H. S. (2005). 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*, 31(7), 551–569.
- 45
- 46 Morbach, J., Wiesner, A., & Marquardt, W. (2009). OntoCAPE—A (re) usable ontology for computer-aided process engineering. *Computers & Chemical Engineering*, 33(10), 1546–1556.
- 47
- 48 Morbach, J., Yang, A., & Marquardt, W. (2007). OntoCAPE: A large-scale ontology for chemical process engineering. *Engineering applications of artificial intelligence*, 20(2), 147–161.
- 49
- 50 Nestor, D., O’Malley, L., Quigley, A., Sikora, E., & Thiel, S. (2007). Visualisation of variability in software product line engineering.
- 51
- 52 Niknam, M., & Karshenas, S. (2017). A shared ontology approach to semantic representation of BIM data. *Automation in Construction*, 80, 22–36.
- 53
- 54 Palvia, P., & Nosek, J. T. (1990). An empirical evaluation of system development methodologies. *Information Resources Management Journal (IRMJ)*, 3(3), 23–33.
- 55
- 56 Pinto, H. S., Staab, S., & Tempich, C. (2004). DILIGENT: Towards a fine-grained methodology for distributed, loosely-controlled and evolInG. *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)* (Vol. 110, p. 393).
- 57
- 58 Pohl, K., Böckle, G., & Der Linden, F. J. van. (2005). *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media.
- Romero, A. A., Kaminski, M., Grau, B. C., & Horrocks, I. (2015). Ontology Module Extraction via Datalog Reasoning. *AAAI* (pp. 1410–1416).
- Ruy, F. B., Guizzardi, G., Falbo, R. A., Reginato, C. C., & Santos, V. A. (2017). From reference ontologies to ontology patterns and back. *Data & Knowledge Engineering*.

- 1 Spyns, P., Tang, Y., & Meersman, R. (2008). An ontology engineering methodology for DOGMA. *Applied Ontology*, 3(1-2),
2 13–39.
- 3 Stuckenschmidt, H., & Klein, M. (2003). Integrity and change in modular ontologies. *IJCAI* (pp. 900–908).
- 4 Suárez-Figueroa, M. C. (2010). *NeOn Methodology for building ontology networks: specification, scheduling and reuse*.
5 Informatica.
- 6 Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M. (2015). The NeOn Methodology framework: A scenario-
7 based methodology for ontology development. *Applied ontology*, 10(2), 107–145.
- 8 Sure, Y., Staab, S., & Studer, R. (2004). On-to-knowledge methodology (OTKM). *Handbook on ontologies* (pp. 117–132).
9 Springer.
- 10 Thakker, D., Dimitrova, V., Lau, L., Denaux, R., Karanasios, S., & Yang-Turner, F. (2011). A priori ontology modularisation in
11 ill-defined domains. *Proceedings of the 7th International Conference on Semantic Systems* (pp. 167–170). ACM.
- 12 Vandebussche, P.-Y., Ateazing, G. A., Poveda-Villalón, M., & Vatan, B. (2017). Linked Open Vocabularies (LOV): a
13 gateway to reusable semantic vocabularies on the Web. *Semantic Web*, 8(3), 437–452.
- 14 Wache, H., Voegelé, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). Ontology-based
15 integration of information—a survey of existing approaches. *IJCAI-01 workshop: ontologies and information sharing* (Vol. 2001,
16 pp. 108–117). Citeseer.
- 17