Chapter 5
Semantic Modelling and Acquisition of Engineering Knowledge

Marta Sabou, Olga Kovalenko and Petr Novák

Abstract Ontologies are key Semantic Web technologies (SWTs) that provide means to formally and explicitly represent domain knowledge in terms of key domain concepts and their relations. Therefore, the creation of intelligent engineering applications (IEAs) that rely on SWTs depends on the creation of a suitable ontology that semantically models engineering knowledge and the representation of engineering data in terms of this ontology (i.e., through a knowledge acquisition process). The tasks of semantic modelling and acquisition of engineering knowledge are, however, complex tasks that rely on specialized skills provided by a knowledge engineer and can therefore be daunting for those SWT adopters that do not possess this skill set. This chapter aims to support these SWT adopters by summing up essential knowledge for creating and populating ontologies including: ontology engineering methodologies and methods for assessing the quality of the created ontologies. The chapter provides examples of concrete engineering ontologies, and classifies these engineering ontologies in a framework based on the *Product-Process-Resource* abstraction. The chapter also contains examples of best practices for modelling common situations in the engineering domain using ontology design patterns, and gives an overview of the current tools that engineers ca use to lift engineering data stored in legacy formats (such as, spreadsheets, XML files, and databases, etc.) to a semantic representation.

Keywords Ontology modelling · Ontology engineering methodologies · Ontology evaluation · Classification of engineering ontologies · Ontology design patterns · Ontology population

M. Sabou (✉) · O. Kovalenko · P. Novák
Institute of Software Technology and Interactive Systems, CDL-Flex,
Vienna University of Technology, Vienna, Austria
e-mail: Marta.Sabou@ifs.tuwien.ac.at

O. Kovalenko
e-mail: Olga.Kovalenko@tuwien.ac.at

P. Novák
e-mail: Petr.Novak@tuwien.ac.at

5.1 Introduction

Engineering knowledge is a specific kind of knowledge oriented towards the production of artifacts, and, as such, requires knowledge modelling and representation approaches that differ from other types of knowledge, such as, for example, taxonomical knowledge that is characteristic for the life sciences domain (Sicilia et al. 2009). Ontologies are information artefacts that can be used to explicitly represent such engineering knowledge and as such, they play an important role when creating intelligent engineering applications (IEAs) relying on Semantic Web technologies (SWTs). Chapter 3 provides a detailed insight into ontologies and their role in SWT based solutions.

Yet, the creation of semantic structures (which we refer to as *semantic modelling*) in general, and of ontologies in particular, is a complex process because of a set of factors (Gomez-Perez et al. 2004). First, the creation process needs to lead to a complete and correct representation of the subject domain to the extent required by the intended use of the ontology. This step requires a deep knowledge of the underlying subject domain as well as a good understanding of the way the ontology will be used. Multi-disciplinary engineering settings such as those specific for creating *cyber-physical production systems* (CPPSs), pose a challenge from this perspective because (1) they require knowledge of several engineering disciplines and (2) they are built for only a partially specified set of applications with new applications being added over time. Second, an optimal machine-understandable representation of the engineering knowledge must be chosen that enables taking full advantage of advanced ontology exploitation mechanisms such as querying and reasoning. To support ontology engineers in finding the best conceptualizations for a variety of modelling requirements, the ontology engineering community has distilled a set of *ontology design patterns* (ODP)—modelling best practices applicable to typical conceptualization scenarios (Gangemi and Presutti 2009).

The complexity of the semantic modelling process often hinders the adoption of ontology-based technologies. Adopters of ontology-based solutions from industry are confronted with answering questions which naturally arise while following ontology engineering methodology steps discussed in Sect. 5.2: *What is the process of building an ontology? What kind of knowledge should the ontology contain? What other relevant ontologies exist and can these be reused for the project at hand? How can the quality of an ontology be assessed? If a new ontology must be designed, what are typical modelling issues to be considered and what typical solutions are applied? How can semantic data be created from legacy data sources?* In this chapter, we aim to answer such questions. Concretely, we:

- Provide a brief introduction to *ontology engineering methodologies* (Sect. 5.2) and the main techniques for *ontology evaluation* (Sect. 5.3).
- Synthesize the types of engineering knowledge that are most often captured by ontologies (Sect. 5.4) and propose an *ontology classification framework* based on ontology content.

- Provide an overview of a set of *engineering ontologies* and classify those according to their content (Sect. 5.5).
- Exemplify a set of of *ontology design patterns* and their use for the semantic modelling of engineering knowledge (Sect. 5.6).
- Provide an overview of techniques and tools for transforming engineering data from legacy data formats (spreadsheets, *Extensible Markup Language*[1] (XML), databases) into Semantic Web representations (Sect. 5.7).

Section 5.8 concludes the chapter with a summary of its content and future work.

5.2 Ontology Engineering Methodologies

A wealth of methodologies exists for guiding the creation of ontologies some dating back to as early as 1990 (Lenat and Guha 1990). Overviews of these methodologies are available in (Corcho et al. 2003; Suárez-Figueroa 2012; Poveda-Villalón 2016). In this section, we summarize the most representative ontology engineering methodologies for supporting ontology adopters in choosing the best-suited methodology for their context. Readers not familiar with the notion of ontologies might wish to consult Chap. 3 for foundations about ontologies.

From the proposed methodologies, the ontology creation guideline of Noy and McGuinness (2001) captures the essential steps of any ontology building process and, thanks to its many examples, it offers an excellent resource for developing one's first ontologies. Hereby, we summarize the main steps indicated in this guideline to give an intuitive overview of a typical ontology creation activity:

1. **Determine the domain and scope of the ontology**. This step defines the domain that the ontology will cover (e.g., mechanical engineering, functional testing), the expected usage of the ontology in terms of the questions that the ontology should be able to answer (also known as, *competency questions*), and the expected stakeholders involved with the ontology (maintainers, users). All these aspects are crucial for focusing the development of the ontology. The competency questions provide a means to define the scope of the ontology, and also, to validate the ontology during or at the end of the development cycle.
2. **Consider reusing existing ontologies**. One of the recommended practices in ontology development is to reuse (parts of) existing ontologies. The promise of reuse is that the developers will be able to build ontologies faster and with less effort, and that the applications that make use of these ontologies will interoperate easier. The practice of reuse is particularly useful in enterprises thus ensuring enterprise-level knowledge reuse, especially if the reused parts fit already well for the new purpose.

[1]XML: https://www.w3.org/TR/REC-xml/.

3. **Enumerate important terms in the ontology**. After defining the scope of the ontology (Step 1), the development of the ontology starts with collecting the most important terms by using the competency questions as input. The competency questions help identify key concepts, their properties and relationships that hold in the domain. This term list acts as an input to steps 4 and 5 for creating the ontology's class hierarchy and properties.
4. **Define the classes and the class hierarchy**. To create a class hierarchy, ontology engineers can choose one of three approaches. (1) In a *top-down approach*, the hierarchy is constructed starting from the most generic concepts (top concepts) and then creating the more specialized concepts. For example, in the mechanical engineering domain, the concept Device is created first, and then sub-concepts are created for the various types of devices (see for example, the ontology snippet in Fig. 3.3). (2) The opposite of the top-down approach consists in starting with the most specific classes of a domain (e.g., the various types of devices), and then grouping these into more generic classes. This approach is known as the *bottom-up approach*. (3) A *combination* approach is taken that combines the top-down and bottom-up approaches, and intertwines these, as most suitable for the ontology expert.
5. **Define the properties of classes**. Another important step is describing the declared classes in terms of their properties, including both their characteristics or relations to other classes. For example, the class `Device` has characteristics such as *weight* or *maximum allowed working hours*. It is connected to the class `Supplier` with the `hasSupplier` property.
6. **Define the property constraints**. This step envisions further enriching the ontology model by defining characteristics of its properties in more detail. These include the domain and range of properties, as well as any cardinality and value-based constraints, which specify the number and type of values that the property takes. Chapter 3 exemplifies how such constraints can be represented using primitives of the *Web Ontology Language* (OWL).
7. **Create instances**. In this step, users populate the classes in the ontology with concrete entities, that exist in the domain, a.k.a, instances or individuals. For example, a concrete device is represented as an instance of the class `Device`, and a user fills its properties with concrete values (e.g., `weight = 100 kg`). In many cases, the actual instances of an ontology are not part of the ontology itself, but are rather acquired using automatic data transformation processes, such as those described in Sect. 5.7.

The **On-to-Knowledge methodology** (Staab et al. 2001) takes a broader view of the ontology life cycle than (Noy and McGuinness 2001) and focuses on an *application-oriented* development of ontologies in the context of creating Knowledge Management applications. Therefore, its steps consider not only the development of the ontology per se, but also that of the application that will make use of the ontology. The methodology consists of five stages:

1. **Feasibility study** identifies problems and opportunity areas from an organizational perspective, and selects the most promising focus areas and suitable solutions.
2. **Ontology kickoff** produces an ontology requirements specification document, which identifies the ontology's goal, domain and scope. This document also identifies the applications that will make use of the ontology, the knowledge sources from which the ontology could be built (e.g., domain experts, relevant documents) and the competency questions it should provide answers to. This phase also covers the identification of ontologies that could be reused and therefore spans steps 1 and 2 in the Noy and McGuinness' guidelines (2001).
3. **Refinement** refers to the development of a mature and application-oriented ontology, and therefore covers activities such as described by Noy and McGuinness' (2001) steps 3–7. Similarly, it envisions stages for eliciting knowledge from domain experts at an epistemological level (i.e., enumerating important terms and their relations) and then formalizing this knowledge in terms of a formal knowledge-representation language such as OWL.
4. **Evaluation** focuses on assessing the quality of the output ontology both in terms of (1) satisfying the *ontology requirements document,* as well as (2) providing the desired functionality as part of the applications that make use of the ontology. Feedback from the requirements and application-oriented evaluations are used to modify the ontology accordingly. Therefore, *On-to-Knowledge* envisions a feedback loop between the *Evaluation* and *Refinement* stages.
5. **Maintenance** covers activities of updating and versioning the ontology to reflect changes in its environment (e.g., new user requirements, supporting new applications, changes in the engineering knowledge).

The **METHONTOLOGY** methodology (Fernández-López et al. 1997), (Blázquez et al. 1998) goes beyond the *On-to-Knowledge* methodology in terms of breadth and distinguishes between management, development, and support activities. Management activities include scheduling, control, and quality assurance. Development covers the entire life cycle of an ontology, and similarly to *On-to-Knowledge* ranges from feasibility study, through development (specification, conceptualization, formalization, and implementation) and post-developmental use to maintenance. Support activities include knowledge acquisition, evaluation, documentation, merging and alignment.

Proposed in 2010, the **NeOn Methodology** (Suárez-Figueroa 2012; 2015) recognized the need of catering for diverse ontology-engineering scenarios, as opposed to prescribing a generic process for building ontologies, as was the case for the previously-mentioned methodologies. The NeOn methodology identifies nine different scenarios that can be followed when building ontologies. The base scenario refers to those cases when ontologies are created from scratch without reusing existing resources. This scenario typically includes stages for requirements specification (identifying the goal, scope, and relevant competency questions for the desired ontologies), scheduling of the ontology creation task, conceptualizing the knowledge in relevant models, and then formalizing and implementing this knowledge in terms of formal representation languages such as OWL, similarly to the steps envisioned by the guidelines of Noy and McGuinness (2001).

Ideally, the effort needed for the base scenario could be reduced by bootstrapping the ontology-creation activity through the reuse of ontological or non-ontological resources. Scenario 2 of this methodology envisions creating an ontology by re-engineering non-ontological resources such as classification schemes, thesauri, lexicons, and folksonomies. Scenario 3 covers cases when an ontology can be re-used instead of building a new ontology. In some cases, it might be necessary to re-engineer an existing ontology (Scenario 4), to merge several suitable ontologies into one (Scenario 5), to reuse a set of ontologies that are merged and also re-engineered (Scenario 6) or to restructure an existing ontology (Scenario 8). Scenario 7 envisions creating ontologies by reusing *Ontology Design Patterns*, which are high-quality ontology modelling solutions. Scenario 9 refers to creating a new ontology by localizing an ontology to better fit other language or culture communities.

Several ontology engineering methodologies considered scenarios of collaborative ontology development. Representative for this set of methodologies is **DILIGENT** (Pinto et al. 2004), which has a strong focus on managing the collaborative evolution of an ontology. In a first stage, a small and consensual ontology is built by the key stakeholders in the ontology creation process (*Build* phase) by following one of the classical ontology development methodologies described above. This version of the ontology is distributed among its users, who can modify their local ontology copy according to their needs (*Local adaptation* phase). In a follow-up *Analysis* phase, an *ontology control board* discusses the changes made locally by users as well as their arguments for introducing their changes. Based on these inputs and their analysis, in a *Revision* phase, the shared ontology is revised and extended to cover an agreed set of changes. During *Local update*, the users update their local ontology to the new version of the shared ontology.

In summary, ontology-engineering methodologies exist for covering a wide range of settings: focusing on ontology creation per se; considering the lifecycle of the ontology-based applications as well as the organizational context, ontology creation supported by various reuse activities, and collaborative ontology development. These methodologies can provide ample guidelines to practitioners, who wish to create ontologies and ontology-based applications. An important step in ontology engineering is ontology evaluation and, therefore, we focus on this specific aspect in Sect. 5.3.

5.3 Ontology Evaluation

Ontology evaluation is *"the activity of checking the technical quality of an ontology against a frame of reference"* (Sabou and Fernandez 2012). Ontology evaluation is also important during other ontology-related activities, such as: the evolution of an ontology; the selection of an ontology for reuse, or during the process of modularizing an ontology (for assuring the high quality of the resulting modules).

The ontology evaluation goal determines the key aspects of the ontology to be assessed and determines the evaluation approaches and measures to be applied. According to (Sabou and Fernandez 2012), the most frequently evaluated ontology aspects are:

1. *Domain coverage—Does the ontology cover a topic domain?* The goal is to assess the extent to which an ontology contains the knowledge necessary to describe a given (aspect of a) domain. For example, one could assess how well an ontology covers terms and their relations when describing *mechanical engineering* aspects of certain types of production systems. This assessment is important to be made both during the development of an ontology and during the selection of an already built ontology. Typically, evaluations with this goal involve the comparison of the ontology to frames of references such as a reference (i.e., gold standard) ontology (Maedche and Staab 2002), or various data sets that are representative for the domain. These datasets can be user-defined terms (Alani et al. 2006; Fernandez et al. 2006), folksonomy tag sets (Cantador et al. 2007), or representative document corpora (Brewster et al. 2004). Typical evaluation measures are similarity measures that compare two ontologies at lexical (i.e., average string matches between the set of gold standard terms and the set of ontology terms), taxonomic, or relation level (Maedche and Staab 2002).
2. *Quality of the modelling.* The evaluation can either focus on the quality of the design and development process (*Does the ontology development process comply with ontology modelling best practices?*) or on the quality of the resulting ontology (*Is the ontology model correct?*). The quality of the ontology model can be assessed using a wide range of approaches focusing on logical correctness or syntactic, structural, and semantic quality. Logical correctness (e.g., logical consistency) is automatically assessed by reasoners. Other aspects, such as syntactic quality, require human judgment and therefore rely on approaches involving human assessment (Burton-Jones et al. 2005). Last but not least, semantic quality can be assessed with metrics such as *essence* (assess if an entity is true in every possible world) or *unity* (recognizes all the parts that form an individual entity) (Guarino and Welty 2004). The OOPS! (OntOlogy Pitfall Scanner!) tool[2] provides online support for verifying ontology modelling quality (Poveda-Villalón et al. 2014).
3. *Suitability for an application/task—Is the ontology suitable for a specific application/task*? While the previous evaluation goals focus on assessing the ontology quality in general, here the aim is to assess to what extent an ontology is suitable for use within a concrete application or for a certain task (e.g., semantic search, question answering). Different applications/tasks might require ontologies with diverse characteristics. For example, for applications that use

[2]OOPS! Tool: http://oops.linkeddata.es/.

ontologies to support natural language processing tasks such as semantic search, domain coverage is often more important than logical correctness. Therefore, measuring ontology quality generically is not enough to predict how well the ontology (developed or reused) will support an application or a task. Task-centric evaluations help assessing suitability for a task or application (Porzel and Malaka 2004; Fernandez et al. 2009). The typical approach here is to measure ontology quality indirectly and as an approximation of an expected application/task performance. For example, for a semantic-search system the precision and/or recall of the system obtained when using different ontologies will be an indication of the ontology's suitability for that task. The best ontology is the one that leads to the best task performance.

4. *Adoption and use—Has the ontology been reused (imported) as part of other ontologies? How did others rate the ontology?* (Cantador et al. 2007) When selecting an ontology for reuse, the extent of its adoption is of particular interest. The assumption is that there is a direct correlation between the quality of the ontology and the level of adoption by the community. One approach to assess adoption is analyzing the degree of interlinking between an ontology and other ontologies (e.g., in terms of reused terms or ontology imports). Ontology libraries often offer such statistics (d'Aquin and Noy 2012). Social rating systems have also been used to reflect community-level reuse and evaluation of ontologies (Cantador et al. 2007).

The evaluation process is guided by the evaluator's understanding of what is better and what is worse. In some cases, these boundaries (which we refer to as frame of reference) are clearly defined and tangible (e.g., a reference ontology, a reference alignment), but in other cases, they are weakly defined and may be different from one person to another, or even across evaluation sessions. This often renders ontology evaluation a non-trivial task.

In summary, several approaches and measures exist for evaluating ontologies, and their selection should be derived based on the goal of the evaluation.

5.4 Classification of Engineering Ontologies

The domain of engineering cyber-physical production systems (CPPS) is a broad and complex domain. Building IEAs for this domain will therefore require creating a diverse variety of ontologies covering the many aspects of the domain. When creating such ontologies, two important aspects to consider are:

- *What kinds of knowledge should be covered by the engineering ontology?* This is an important question for determining the scope of the ontology, but that generally, ontology engineers (i.e., Semantic Web practitioners), who have little knowledge of this complex domain, find difficult to answer.

- *How to find suitable engineering ontologies for reuse?* Ontology reuse is an important step envisioned by all ontology engineering methodologies. Yet, Legat et al. (2014) observe that the reuse of ontologies in the automation domain is difficult, in part because of a lack of a principled way to create and classify ontologies in an use-case agnostic way.

One approach to alleviate these difficulties in scoping, classifying, and reusing ontologies is the availability of meaningful schemes for classifying the type of knowledge relevant for engineering complex CPPS. Such a scheme could support better scoping ontologies in terms of use-case-independent topics, and could greatly support meaningful selection of relevant ontologies.

In this section, we propose such a classification scheme, which we derived by combining two orthogonal views on the types of engineering knowledge in CPPS. First, the *Product-Process-Resource (PPR)* abstraction (Sect. 5.4.1) provides an intuitive view of the domain of production systems (Schleipen and Drath 2009). Second, we made use of an initial ontology-classification scheme (Sect. 5.4.2), proposed by Legat et al. (2014). This scheme is much more oriented on structure than the PPR view, because it considers the different physical and logical views on the elements of a production system. The proposed categorization framework for engineering ontologies, which combines these two views, will allow practitioners to find easier the ontologies that cover the relevant engineering content for their project. The categorization framework will also be useful to Semantic Web practitioners, who require a better understanding of knowledge needs, when building IEAs and scoping their ontologies.

5.4.1 The Product-Process-Resource Abstraction

To better understand the types of engineering data needed to describe a complex manufacturing plant, we start with the *Product-Process-Resource* abstraction explained in (Schleipen and Drath 2009). The three views of this abstraction are dominant, and of key interest for industry, as described in more detail next.

Product. The intention of any production system is to produce products (e.g., cars, bread). The term “product” refers to the final produced goods as well as to the produced artefact in one of its intermediary stages. A product designer describes a product by its geometric, functional, material, and other relevant characteristics. Depending on the kind of product, the *product structure* can be described with mechanical information, e.g., 3D CAD and kinematics.

Process. To create a product, a set of *production process* steps is applied to manufacture the product from raw materials or semi-finished products. Processes modify intermediate products and create a final product. Example processes are welding, transporting, and filing. The process view is concerned with the set of processes (their parameters, and their chains) needed to create a product from input materials. The production process corresponds to the function description of the production resources, and to the control, e.g., a PLC program. However, the production process can also be described more explicitly, e.g., with GANTT charts or some general form of behaviour description, e.g., state machines.

Resource. Production resources are entities involved in production that provide functions to production processes. Resources include both hardware (e.g., robots, conveyors, machines) and software entities (SCADA systems), and are typically organized hierarchically into a *plant topology*. The properties of production resources are mostly their function capabilities (e.g., welding, transporting, and filing), mechanical information, electrical information, and control-related information; in addition, further technical information can be specified.

There is a strong connection between these three views: a *product* is manipulated by a *resource* as part of a *process*. For example, in the case of a tetra-pack production system described in (Schleipen and Drath 2009), the process *transport* might be linked to a resource *conveyor* and a product *tetra pack*.

Although it is useful to consider these three views in separation, they all contribute to the definition of a *production system*. The intention of any *production system* is to execute *production processes* using *production resources* in order to create *products*. Example production systems are: car body welding shops, combustion-turbine-based power plants or process plants for polyvinyl chloride generation. Therefore, a production system contains a set of interlinked production resources each of them able to execute a certain production process step (drilling, welding, etc.). The production system designer describes a production system by the information given in Fig. 2.6 and provided by the different engineers named in Chap. 2. This information includes:

- topology information describing the hierarchy of production resources;
- geometry and kinematics information;
- network information (electrical, pneumatic, communication, …); and
- control information.

The aim of each production system is the creation of products. Therefore, the *PPR abstraction* follows the Input-Processing-Output principle. Input materials are processed in processes to lead to output products (and waste) by exploiting resources.

5.4.2 A Classification Scheme for Engineering Ontologies

While the *PPR* view provides a concise description of the main concepts in the production systems engineering domain, which could be covered by ontologies, Legat et al. (2014) propose a complementary view on typical ontology content which was developed in a bottom-up fashion during a survey and classification of already available automation ontologies. Concretely, Legat et al. (2014) observe that the reuse of ontologies in the automation domain is difficult. One solution that they propose is that ontology reuse could be improved by modularizing ontologies along topics that are independent of a specific use case. The module structure they propose, provides good insight into the types of information that are relevant when describing CPPS.

In the following, we discuss the main categories of information identified by Legat et al. (2014) and how these relate to the PPR abstraction. By combining these two orthogonal views we obtain a broader classification scheme of engineering knowledge that connects the well-accepted topics from industry (PPR) with use-case-agnostic topics typically covered by ontologies. This framework is depicted in Table 5.1, where columns correspond to PPR concepts and rows represent to the main categories of information according to Legat et al. (2014), namely:

- **Physical objects** description. This description category refers to the categorization of available objects, such as equipment in a plant (e.g., sensors, actuators), product parts that need to be assembled. The properties, functionalities, and structures of these objects are covered by other ontologies. As such, this category in Legat et al.'s classification covers both products and production

Table 5.1 Ontology classification framework combining PPR concepts (columns) and Legat et al.'s modular ontology view (rows)

	Product	Production process	Production resource
Physical objects	Ont. of product types, *OntoCAPE, eClassOWL*	*OntoCAPE, ISO 15926*	Ont. of resource types, *OntoCAPE, CCO, AMLO, ManufOnto, eClassOWL, ISO 15926, AutomOnto*
Structure	Ont. of product structure, *OntoCAPE*	NF	Ont. of resource structures, *OntoCAPE, ManufOnto, EquipOnt, CCO, AMLO, ISO 15926, AutomOnto*
Functionality	NF	Ont. of production process types, *OntoCAPE, ISO 15926, ManufOnto*	Ont. of production resource capabilities (skills), *AMLO, ManufOnto, EquipOnt, ISO 15926*
Process	NF	Ont. of production process structures, *OntoCAPE, ISO 15926, ManufOnto*	*ManufOnto*
Materials	Ont. of bills of materials, *eClassOWL*	NF	NF
Observations, measurements	NF	Ont. of process states and its observability, *SSN, OntoCAPE, ISO 15926*	Ont. of resource states, *SSN, AutomOnto*
Quantities, dimensions, units	Ont. of product characteristics, *eClassOWL*	Ont. of production processes characteristics, *OntoCAPE, ISO 15926*	Ont. of production resource characteristics, *ManufOnto, CCO, SSN, AutomOnto*

Example ontologies added in *italics*. NF = not feasible resources in PPR, meaning that ontologies that describe types of products or types production resources can be considered, from Legat et al.'s perspective, as ontologies describing physical objects (see Table 5.1).

- **Structures** description. Ontologies are also needed to describe various structural information about how the physical objects are composed. For example, one can describe how products need to be assembled, how production resources are structured, or how production systems (plants) are constructed. Structure-related information can be conveyed in different ways, according to Legat et al. (2014): (1) **interface-based composition** describes the capabilities expected from an interface and can enable reasoning tasks about the correctness of a system's structure; (2) **containment** hierarchies are a well accepted and frequently occurring organizational paradigm in mechatronic engineering settings.
- **Functionality** describes the behaviour of devices (or organizational units), i.e., what these elements can do in terms of contributions to a production process. Functionality descriptions are characterized by their properties, parameters and constraints, for example in terms of materials they can be applied on. Functionality descriptions are relevant both for individual production resources (e.g., a description of the function fulfilled by a resources, such as filling or drilling) as well as by production processes (that is, a function achieved by a process).
- **Process** descriptions are closely related to production resource functionality; they enable describing the composition of functionalities into obtaining functionalities that are more complex. Examples here are workflow patterns, such as the sequential or parallel execution of functions. Process ontologies provide useful primitives for describing the structure of production processes, for example, the workflow structure in which its steps are organized.
- **Material** descriptions contain collections of materials used within a production process and their hierarchical organization (e.g., wood, iron). Ontologies describing *bills of materials* needed for creating a product fit this category.
- **Observations and measurements** capture data produced during a system's operation, and as such differ in nature from the previously mentioned knowledge categories. Such information is essential to capture physical changes that happen in the physical world in which a CPPS operates. Relevant types of ontologies would be those describing the state of production processes or production resources.
- **Physical quantities, dimensions, and units** provide auxiliary information for describing aspects of most of the knowledge categories mentioned above, related to temporal aspects, weight, spatial dimensions (e.g., length). Such ontologies can be used to describe the various characteristics of products, production processes, production resources, and production systems.

From Table 5.1 it becomes evident that ontologies in engineering vary widely and span different aspects of different engineering concepts. Beyond the identified types of ontologies, there are several engineering-discipline-specific ontologies in place. These ontologies usually cover only the concepts relevant within a special application case, such as production system simulation at resource level, or, they may cover special engineering discipline knowledge, such as an automation glossary.[3] In Sect. 5.5 we provide examples of engineering ontologies, and classify them according to the classification scheme proposed in this section.

5.5 Examples of Engineering Ontologies

In this section, we provide an overview of engineering ontologies and classify them according to the classification scheme defined in Sect. 5.4.2. We start by describing ontologies published in literature, and conclude with two ontologies created at the *CDL-Flex* Laboratory[4]: the *AutomationML Ontology (AMLO)* and the *Common Concepts* ontology (CCO), in Sects. 5.5.1 and 5.5.2 respectively.

OntoCAPE[5] is an ontology for supporting *computer-aided process engineering* (CAPE). It was designed at the RWTH Aachen University, and it is discussed in numerous publications in details (Morbach et al. 2009). It has a modular structure consisting of 60 + OWL files (with the meta-model, it has more than 80 files).

OntoCAPE has a layered logical design spanning from foundational to domain-specific layers. The lowest level is called the *application-specific layer,* and it contains application-specific knowledge. The next level is the *application-oriented layer*, which describes the plant and process control equipment. This level also includes the view on the particular technology from the process point of view. The third level is the *conceptual layer*, which provides supporting concepts for modelling of processes, materials, and other representations and characteristics needed for modelling of processes. The top level is called the *upper layer,* and provides expressive means for representing networks, coordinate systems, and others. *OntoCAPE* is defined through a meta model, which is also denoted as the *meta layer* of *OntoCAPE*. The meta layer is represented as a stand-alone OWL ontology, which provides foundations for meta-modelling structures and other fundamental concepts.

In terms of the terminology introduced in Chap. 3, *OntoCAPE* combines ontology layers that include domain ontologies, generic ontologies (i.e., the conceptual layer) and foundational ontologies (i.e., the meta-layer). *OntoCAPE* is highly axiomatized. Because of its breadth, *OntoCAPE* can be classified in our classification scheme (Sect. 5.4.2) under several types of ontologies. From the PPR perspective, it entirely covers the production process criterion. *OntoCAPE* also addresses all other criteria, but focuses mainly on physical objects and structure levels, according to the classification by Legat et al. (2014).

[3]Automation glossary: http://ai.ifak.eu/glossar2/.

[4]CDL-Flex Laboratory: http://cdl.ifs.tuwien.ac.at/.

[5]*OntoCAPE* is available at: https://www.avt.rwth-aachen.de/AVT/index.php?id=730.

ISO 15926[6] is a complex standard dealing with industrial automation systems and integration (ISO 15926). Although it has been originally intended for the oil industry, its ideas and approaches are general and usable also in other domains. The main part of the standard is Part 2, dealing with description of objects and activities during various stages of the plant life cycle. The ontology includes diverse views on the process plant depending on the involved engineering disciplines. The original version of the standard (Parts 2–4) use the EXPRESS language, which was difficult to use and has a limited tool support. Hence, the standard was enhanced with Parts 7–10, relying on the OWL language. The OWL representation of Part 2 is available online.

The ISO 15926 defines an ontology that is generic, heavyweight and covers mainly information about production processes. It also includes information about production resources and their evolution, especially in terms of physical objects, structure, functionality, and materials.

ManufOnto. Alsafi and Vyatkin (2010) developed the Manufacturing Ontology to model modular manufacturing systems. The ontology (*ManufOnto* in Table 5.1) describes the machinery and operations in a semantic way, in order to facilitate flexibility and agility of manufacturing. The ontology is not available online for reuse, but nevertheless provides an example of the types of ontologies designed for manufacturing systems. The top (e.g., most generic) part discussed in (Alsafi and Vyatkin 2010) includes 29 concepts and 39 properties. The main idea reflected in the ontology is to separate and interrelate (i) the required operations, (ii) the physical machinery performing the required operations, and (iii) the control of the machinery. Based on the available description, we classify it as a lightweight, domain ontology, which covers aspects such as: resource types, process types, and products from the perspective of processes, functionalities, and structures.

EquipOnt. Lohse et al. (2006) describe the *Equipment Ontology, EquipOnt,* designed to support *Reconfigurable Assembly Systems* (RAS), i.e., systems that allow configuring and reconfiguring assembly systems based on changing requirements of the assembled product. The authors rely on the function-behaviour-structure paradigm in the design of the ontology, because they consider these three aspects of an equipment essential for their use case (i.e., deciding whether a piece of equipment can be reused in an assembly system). In their view, "functions express the capabilities of a module based on the intention of their designer". The behaviour of an equipment defines its reaction to changes in the environment. The physical model of the equipment represents its structure. The authors adopt a modular ontology design, with different modules being dedicated to covering the three aspects of function, behaviour and structure. The class *Equipment* is a key class in the ontology, and a superclass for different types of equipment such as *Device*, *Unit*, *Cell*, *Workstation*. The internal structure of elements is modelled with the *subComponents* relation (partOf) as well as with elements to describe connections between components. Each *Equipment* is also associated to a

[6]ISO15926 is available at: https://www.posccaesar.org/wiki/ISO15926inOWL.

Function and a *Behaviour* object. Similar to *Equipment*, functions and behaviour are also organized in specialization hierarchies.

The E*quipment Ontology* is not available online. Based on the available description, we classify *EquipOnt* as a lightweight, domain ontology, which covers aspects of production resource structures and capabilities.

eClassOWL[7] (Hepp 2006) is an OWL representation of the *eClass* standard,[8] which is a cross-industry catalogue describing the nature and features of various products and services. *eClass* spans more than 30,000 product and service types and includes over 5,000 product or services properties. The parts of *eClassOWL* related to engineering include: machines or devices (for special applications) (17,000,000); electrical engineering, automation, process control engineering (27,000,000); and automotive technology (28,000,000).

eClassOWL aims to provide semantic representation of the types and properties of products and services while preserving the original semantics of *eCl@ss* as much as possible. Since the *eCl@ss* catalogue was designed from a purchasing manager perspective, it does not have a proper subsumption hierarchy. This is the reason why *eClassOWL* creates two classes for each *eCl@ss* category: a "*generic*" class to represent the actual product or service, and a "*taxonomic*" class with a wider semantics (meaning that something can be related to this category). The intention is to use generic concepts for describing the actual products (e.g., a "machine"), and to use taxonomic concepts to describe something that is not a product, but is related to it (e.g., "machine maintenance").

eClassOWL is tightly connected with GoodRelations ontology, developed for the eCommerce domain and describing such product aspects as demand, prices and delivery options. The GoodRelations metamodel is used as a schema skeleton for *eClassOWL*. Therefore, *eClassOWL* inherits the following property types from the GoodRelations meta-model: a) *quantitative* properties (for product features with a numeric range); b) *qualitative* properties (for product features with predefined value instances); and c) *datatype* properties (used only for a few features with the data-types string, date, time, datetime, or Boolean). The domain of all properties in *eClassOWL* is gr:ProductOrService.

Due to copyright and pricing issues, the current version of the *eClassOWL* ontology represents version 5.1.4 of the standard, and therefore lags behind significantly from the latest version of *eCl@ss* which is 9.0. Nevertheless, the process of extracting an OWL ontology from *eCl@ss* is well documented and available (Hepp 2006). Therefore, it can be potentially reused for extracting the ontology from newer versions of the standard. *eClassOWL* is a large, lightweight, domain ontology which covers various aspects of product types and characteristics.

The **Semantic Sensor Network (SSN) ontology**[9] was developed by W3C's Semantic Sensor Networks Incubator Group (SSN-XG) (Compton et al. 2012).

---

[7]eCl@ssOWL: http://www.ebusiness-unibw.org/ontologies/eclass/5.1.4/eclass_514en.zip.

[8]eCl@ss standard: http://www.eclass.de/eclasscontent/standard/overview.html.en.

[9]SSN Ontology: www.w3.org/2005/Incubator/ssn/ssnx/ssn.owl.

The main motivation for this group was to develop an ontology, which captures (a) the event based nature of sensors and sensor networks, for the cases in which the temporal and spatial relationships need to be taken into account; and (b) complex physical constraints (e.g., limited power availability, limited memory, variable data quality, and loose connectivity) that need to be considered for effective reasoning and inference.

The SSN ontology describes sensors, their observations, and related concepts in a general way. This means that no domain-specific information is given: domain semantics, specific units of measurement, etc., can be included, if necessary, via OWL imports, while instantiating the SSN ontology for a particular domain. The SSN ontology focuses on the description of the physical and processing structure of sensors. "Sensors" are not limited simply to physical sensing devices. Rather a sensor is anything that can estimate or calculate the value of some phenomenon. Thus, a device or computational process, or combination of those can be considered as a sensor. A "sensor" in the ontology links together *what it measures* (the domain phenomena), the *physical sensor* (the device) and *its functions and processing* (the models). Therefore, depending on the application at hand, the SSN ontology allows focusing on different perspectives: (a) a sensor perspective (what senses, how it senses, and what is sensed); (b) a data or observation perspective; (c) a system perspective; and (d) a feature and property perspective.

The SSN ontology is a generic, heavyweight ontology. It has a careful ontological design. For example, it is aligned with the *DOLCE Ultra Lite* upper ontology to facilitate its usage with other ontologies or linked data resources developed elsewhere. It is also based on the *Stimulus-Sensor-Observation* Ontology Design Pattern[10] introduced by (Janowicz and Compton 2010). The SSN ontology is best suited to describe process states and their observability, as well as resource states.

The **Automation Ontology** captures knowledge about industrial plants and their automation systems to support engineering of simulation models (*AutomOnto* in Table 5.1). It has been presented in (Novák et al. 2015). The automation ontology has a mechatronic nature and provides support for simulation model design and integration.

The automation ontology covers four domains and their mappings: a real plant domain, a variable domain, a parameter domain, and a simulation domain. The real plant domain represents the topology of a real system, i.e., it includes physical devices and their connections. Each real device can have assigned one or more parameters and can have input and output variables. Both parameters and variables are formalized in their respective domains. Parameters are considered as physical properties representing device features, such as size or length and other characteristics. They are constant values (i.e., independent on time). On the contrary, variables annotate process variables, inputs, and outputs. A tag can be assigned to

[10]The Stimulus-Sensor-Observation ODP: http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/#The_Stimulus-Sensor-Observation_Ontology_Design_Pattern.

each variable, which is a unique name for the variable shared among automation system tools and algorithms. Values of variables/tags are considered as time-series of physical quantities that are either measured by sensors in the real plant or exported by software parts of automation systems. Finally, the simulation domain is focused on supporting the engineering process of simulation models, which approximate the behavior of the real plant. This is the reason why the simulation domain and the real plant domain are mapped, which is useful for redesigning and reusing simulation artifacts. The simulation artifacts annotated with the automation ontology are simulation modules, which are parts of specific simulation models, or simulation blocks, which are the smallest artifacts supported in this formalization. In addition, the ontology includes expressiveness for representing how all these entities are hierarchically organized and how they correspond to each other. The simulation domain is mapped to parameters and variables/tags as well to support efficient interface description and configuration. Further details about this ontology from an application perspective can be found in Chap. 10.

5.5.1 The AutomationML Ontology

AutomationML (AML) is an open, XML-based data exchange format developed to support exchange of engineering data within the engineering process in production systems (Drath 2010). AML includes information about system topology, geometry, kinematics, and control behaviour. For more details about AML one can check the specification (IEC 62714). The AML representation is based on the four main concepts of CAEX (Computer Aided Engineering Exchange), a data format for storing hierarchical object information, such as the hierarchical architecture of a plant (IEC 62424 2008):

- The *RoleClassLibrary* allows specifying the vendor-independent requirements (i.e., the required properties) for *production-system-equipment objects*. A *role class* describes a physical or logical object as an abstraction of a concrete technical realization (e.g., a robot or a motor). In this way, a role class specifies the semantics of an object enabling automatic interpretation by a tool. Additionally, a role class allows defining attributes that are common for an object.
- The *SystemUnitClassLibrary* allows specifying the capabilities of solution equipment objects that can be matched with the requirements of objects defined with the role classes. A *system unit* class describes a physical or logical object including the concrete technical realization and internal architecture. For example, a System Unit Class *KR1000* that matches with the role class *robot* may describe attributes of the *KUKA KR 1000,* which has a payload of 1,000 kg. Thereby *system unit* classes form a multi-level hierarchy of vendor-specific objects that can be instantiated within the *InstanceHierachy*.
- The *InterfaceClassLibrary* defines a complete set of interfaces required to describe a plant model. An *interface class* can be used to define two types of

relations. First, it can define relations between the objects of a plant topology (these include all kind of relations, e.g., of mechanical nature or signals and variables related to the PLC code). Second, an interface class can serve as a reference to information stored outside the CAEX file (e.g., a 3D description for a robot).

- The *InstanceHierarchy* contains the plant topology, comprising the definition of a specific equipment for an actual project—the instance data. Therefore, all project participants can refer to the instance hierarchy to define the context for their work tasks and results. The instance hierarchy contains all data including attributes, interfaces, role classes, relations, and references.
- Although facilitating data exchange is already an important improvement, there is still a lack of infrastructure for supporting advanced engineering activities across the disciplines and tools in AutomationML-based projects, e.g., for data linking, change propagation across connected datasets, data analysis and consistency checking. In order to address this need, we developed an AutomationML ontology. We provide a solution for moving from AutomationML XML files to an OWL representation, as a prerequisite for developing an ontology-based integration and cross-disciplinary analytics on the top of data represented in this format.

Ontology creation process. As a starting point for the ontology creation, we took the available *XML Schema Definitions* (XSDs) providing the AutomationML language definitions in machine-readable and structured form. The actual ontology-building process is divided into two main steps. In the first step, we performed an *automatic transformation* from XML Schema to OWL using Apache Jena[11] to obtain an initial ontology draft. Here our goal was to provide an ontology draft, that is equivalent to the available XSDs, and that will allow to build efficient tool support. In the second step, we enhanced the draft ontology with additional axioms that would reflect the domain knowledge, but which was not available in the XSD. We also optimized the draft ontology obtained in the first step to reflect more accurately the AutomationML specification, and to use more efficiently the graph-based nature of ontologies. In particular, we replaced the string values of some properties (storing the path expressions to AutomationML structural elements) with relations, which actually refer to those elements.

Using the automatic conversion of the XML schema document into OWL allowed us to rapidly bootstrap the ontology. The final ontology directly matches the original XSD structure, which makes the data transformation from the original AML format to OWL straight-forward. Figure 5.1 presents the core concepts of the AutomationML ontology[12] derived from the AutomationML XSD schema. The AutomationML ontology depicts the inheritance taxonomy of the main concepts as well as selected containment (part-of) relationships.

[11]Apache Jena: https://jena.apache.org.

[12]AutomationML Ontology: http://data.ifs.tuwien.ac.at/aml/ontology#.

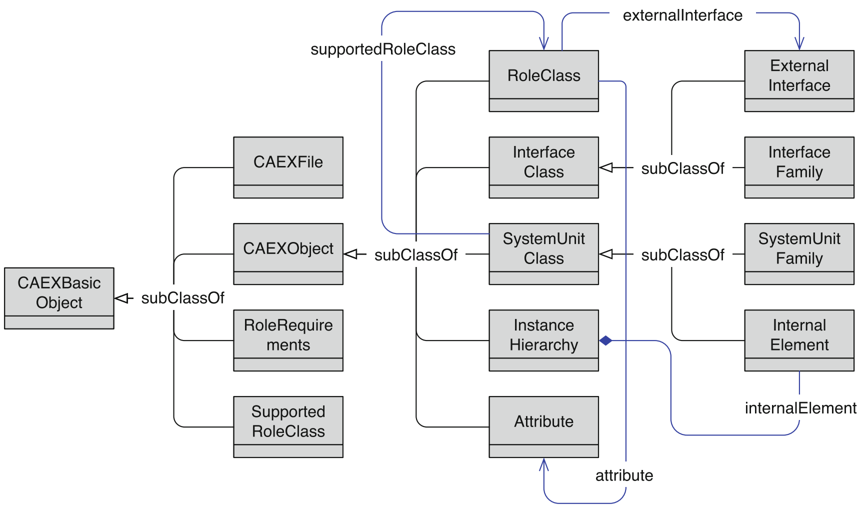

Fig. 5.1 AutomationML ontology: core concepts and their relations

The ontology concepts describe various aspects of production plant design. *RoleClass* represents production plant elements by describing vendor-agnostic functionality of the plant components, e.g., a robot or a motor. *SystemUnitClass* is used for storing vendor-specific plant elements, i.e., physical plant “building blocks” that can be ordered from a vendor and used for plant implementation. The instantiation of system units takes place in the *InstanceHierarchy*. *InstanceHierarchy* stores current plant system, putting the defined *InternalElements* in a hierarchical structure. *InternalElement* describes a specific element of the actual designed plant, including attributes, interfaces, relations, and references. *InterfaceClass* encodes the syntactical and semantic specification of user specific interfaces. Detailed description of the engineering elements can be given by defining various *Attribute*s, e.g., material, weight, maximum power consumption. Attributes can be assigned to interfaces, roles, system units and internal elements. The AutomationML ontology also contains other auxiliary concepts (not shown in Fig. 5.1) covering additional information about the tool that was used to create a specific AutomationML file (such as the name of the exporting software tool, its vendor, version and release information), or a project identifier and title.

The AutomationML ontology supports the full representation of the content of any AutomationML file. As a result, Semantic Web tools and technologies may be applied on AutomationML data as well. For example, a semantic representation of AutomationML files is a pre-requisite for providing methods that semantically integrate those files, and allow advanced analytics on the integrated datasets. An example of such a tool is the AutomationML Analyzer (Sabou et al. 2016).

In summary, the **AutomationML ontology** is a lightweight domain ontology, capturing the structural elements of the AutomationML standard. The ontology covers, therefore, elements of production plant design, such as vendor-specific information about the devices used as building blocks for the plant; vendor-agnostic functions of the plant components; containment relations between the plant elements; and interfaces between the elements and references to other files or engineering objects.

5.5.2 Common Concepts Ontology

The *Common Concepts Ontology* (CCO)[13] aims to capture the most essential concepts necessary to ensure communication between stakeholders involved in the engineering of complex mechatronic systems. This ontology captures high-level concepts in the mechatronic domain, and it can be used to bridge the conceptual gap among engineers from different engineering domains and, between engineers and project managers. As such, the CCO ontology provides a useful approach to foster communication across the diverse stakeholder groups by capturing their shared understanding of the engineering domain.

Besides its value in supporting human communication, the CCO is an information model that acts as a key element of the *Engineering Knowledge Base* (EKB) described in Chap. 4. Namely, it is used as a *global ontology* to which concepts from *local ontologies* are mapped in settings where project-level data integration and access is needed. In such settings, IEAs built to explore the integrated data should answer a variety of competency questions, such as: questions about the project and various engineering roles; questions about engineering objects created by different engineering disciplines and their interconnection. Therefore, the focus of the ontology is primarily on describing the product of the engineering process, without considering in detail information about engineering processes and resources.

The methodology for engineering the CCO followed the guidelines of Noy and McGuinness (2001), where the knowledge elicitation step involved workshops with various domain experts. CCO was built in the CDL-Flex Laboratory and it was re-used and improved in several engineering projects for building diverse mechatronic objects (e.g., steel mills, hydro power plants). CCO represents a set of concepts that can be reused in other engineering projects of similar nature, especially focusing on the creation of production plants. This ontology might have limited use for different engineering projects than the ones that inspired its derivation. Naturally, projects that reuse this ontology might need to adapt it to meet their own needs and characteristics.

The CCO consists of two major parts. First, as depicted on the left hand side of Fig. 5.2, the ontology contains concepts that describe organizational level aspects. These concepts include the *Project*, the *Customer* for whom the project is

[13]CCO Ontology: http://data.ifs.tuwien.ac.at/engineering/cco.

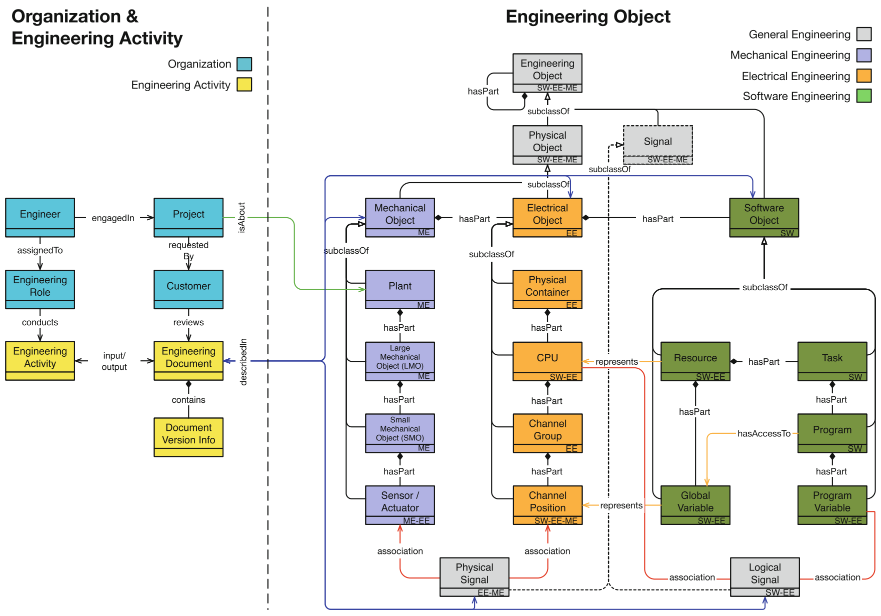

Fig. 5.2 The common concepts ontology

performed, as well as the *Engineers* (and *Engineering Roles*) necessary to realise the project. Engineers conduct *Engineering Activities,* which take as input and create as their output various *Engineering Documents* (e.g., signal lists, design documents). Engineering documents are versioned and reviewed by the customer, thus constituting an important exchange medium between the customer, who requested a project, and the engineering team executing that project.

Second, the CCO describes various *Engineering Objects* created during the engineering project (right-hand side of Fig. 5.2). The ontology identifies different types of engineering objects, such as *Software Objects*, *Mechatronic Objects*, and *Electrical Objects*. Each engineering project leads to the creation of a *Plant*, which is a type of *Mechatronic Object*. The ontology also clarifies the various parts of a mechatronic object, their internal structure and connections among them. To that end, the ontology captures different types of Mechatronic, Electrical and Software objects, and details their internal structure at a high level of abstraction. *Physical Signals* and *Logical Signals* represent the links between engineering objects created by different engineering disciplines and how these diverse components can command or exchange data with each other. In addition to these signals, detailed descriptions of the various mechatronic components are also available as engineering documents (e.g., PLC programs for software objects, or ECAD diagrams for the electrical wiring).

The internal structure of the components captured by the ontology emerged during several projects, and can also be represented using other approaches, such as the AutomationML instance hierarchy or domain-specific structuring standards. For example, in the domain of power plants, the *Kraftwerk-Kennzeichensystem* (KKS)[14] system is used to describe the structure of mechatronic components using a four-level structuring scheme: level 0 denotes the plant, level 1 denotes the function key (e.g., system), level 2 describes an equipment unit key (e.g., *Pumpunit*), and level 3 corresponds to a concrete component (e.g., a drive).

CCO is a lightweight domain ontology, which can be used to describe types and structures of various engineering objects, either at the level of the production system or at the level of production resources.

5.6 Ontology Design Patterns for Engineering

In this section, we discuss more detailed aspects of modelling engineering knowledge. In particular, we identify a non-exhaustive collection of typical modelling needs in engineering, and then exemplify how selected modelling needs can be addressed by various modelling solutions, such as, for example, *Ontology Design Patterns* (ODPs).

To identify representative needs for modelling engineering knowledge in mechatronic systems, we consider elements of *systems engineering* (Stevens et al. 1998), which have been adopted as the basis of mechatronic development process (VDI 2004). More concretely, we consider the elements of the SysML language (OMG 2006), which is a wide-spread modelling language for describing systems. As also discussed in Chap. 12, SysML distinguishes between a component's *definition* and its *usage*. A system's definition consists of the declaration of parts and their connections as well as the declaration of constraints on parts; e.g., *each car has exactly 1 engine*. Usage related aspects refer to the role that a component plays as part of another component. Accordingly, important modelling scenarios in system's engineering include:

- *Modelling Part-whole relations*. Legat et al. (2014) observe that *containment* hierarchies are a well-accepted and frequently occurring organizational paradigm from modelling part-whole relations in mechatronic engineering settings.
- *Modelling connections between components*. Legat et al. (2014) observe that *interface-based composition* describes the capabilities expected from an interface and can enable reasoning tasks about the correctness of a system's structure.
- *Modelling component roles*. Component roles refer to their functions and behaviour that they play in the system.

In what follows, we discuss modelling approaches for part-whole relations and connections between components.

[14]KKS System: https://de.wikipedia.org/wiki/Kraftwerk-Kennzeichensystem.

Modelling Part-Whole Relations. Expressing part-whole relations is an important aspect in several domains, most prominently in engineering and life sciences. Typical use cases involve the creation of inventory applications that, given an engineering object, could report all its parts and the subparts of those; or applications where the composition of an engineering object is explored one level at a time (i.e., only the direct components of the object are shown at any time).

Mereology (or mereotopology) refers to the study of part-whole relations and has been extensively investigated (Artale et al. 1996; Odell 1994; Winston et al. 1987). However, ontology modelling languages, such as RDF(S) and OWL, do not provide built-in primitives for modelling part-whole relations as they do, for example, for modelling subsumption hierarchies (e.g., `rdfs:subClassOf`). There are, however, several approaches to model different types of part-whole relations. We hereby provide an overview of some modelling options.

In Chap. 12, the authors report on the *Components* ontology (depicted in Fig. 12.2), which was designed for capturing part-whole relations in a use case from the car manufacturing industry. The ontology describes `Components` in general and distinguishes between `Composite Components` and `Atomic Components`. A non-transitive `hasPart` relation is introduced to represent *direct* subparts of a component. This modelling caters for use cases, where it is sufficient to retrieve the direct parts of a given component. Inventory-type applications, which should recursively return all the parts of subparts, require the use of a *transitiv*e relation `hasSubpart`. As discussed in Chap. 3, a transitive property satisfying equation Eq. 5.1 supports the computation of the transitive closure of all parts.

$$hasSubPart(A, B) \wedge hasSubPart(B, C) \vDash hasSubPart(A, C) \tag{5.1}$$

The *Component* ontology also declares an inverse relation for `hasPart`, namely `isPartOf`. Note that inverse relations can significantly slow description logic reasoners, and therefore they should be used with care (Rector and Welty 2005).

The ontology engineering community has identified generic modelling approaches for common modelling problems, such as part-whole relations. The ontologydesignpatterns.org is a community-curated portal,[15] which contains *Ontology Design Patterns* (ODP) for *meronomy* and other common situations. This portal recommends three ODPs for modelling part-whole relations:

- The PartOf ODP[16] allows modelling of part-whole relations in a transitive fashion. To that end, it introduces the `hasPart` transitive relation between two `Entities`, as well as its inverse, `isPartOf` (also transitive).
- The Componency ODP[17] is a specialization of the *PartOf* ODP, and offers a solution for modelling part-whole relations in such a way that a distinction can be made between direct and non-direct (i.e., transitively-assessed) parts of an

[15]ODP community portal: http://ontologydesignpatterns.org/.

[16]PartOf ODP: http://ontologydesignpatterns.org/wiki/Submissions:PartOf.

[17]Componency ODP: http://ontologydesignpatterns.org/wiki/Submissions:Componency.

Object. The difference from the *PartOf* ODP is the introduction of the non-transitive `hasComponent` relation as a subproperty of `hasPart`. Note that sub-properties do not inherit the constraints of their super-properties, and therefore `hasComponent` is non-transitive. Furthermore, `isComponentOf` is declared as the inverse of `isPartOf`. Similar to the modelling described in Chap. 12, this ODP caters for deducing both direct and indirect parts of an object. Since `isComponentOf` is a sub-Property of `part of`, it is sufficient to assert it between relevant Object instances, as the `part of` relation will be automatically deduced (by virtue of the semantics of `subPropertyOf` construct). Interested readers can further investigate an example of how this pattern is used to model components of a car.[18] This pattern corresponds to the basic modelling pattern[19] recommended by Rector and Welty (2005).
- The TimeIndexedPartOf ODP[20] caters for situations, which require modelling of a situation in which an object is part of another for a specified time interval. The pattern relies on *n-ary* relationships to establish time-based part-whole relations between a part, a whole, and a time interval.

Besides the patterns described above, other modelling approaches have also been put forward, but their discussion exceeds the scope of this chapter. For example, Rector and Welty (2005) provide five different modelling patterns, while the modelling of ISO 15926 uses property chains for modelling part-whole relations.[21]

While several modelling solutions are proposed, it is also important to avoid confusing part-whole relations with other relations. Rector and Welty (2005) mention typical confusions with relations such as containment, membership, connections, constituents and `subClassOf`. For example, *constituency* refers to a relation without a clear `part of` relationship (e.g., different types of wood constitute a table) and special ODPs are offered for modelling constituency.[22]

Modelling Connections. As discussed in Chap. 12, the behaviour of a system is determined by interactions between its parts, and such interactions are abstracted to connections representing flows of energy, matter, or signals between components. Therefore, an important aspect in defining a system is declaring connections among its components. Chapter 12 presents an approach for modelling connections as part of the *Connection* ontology depicted in Fig. 12.3. This ontology provides a pattern for modelling connections at a high-level of abstraction for any entity that can have connections (which is conceptualized as `TopologicalIndividual`). For example, systems and individual system components can be considered as types of `TopologicalIndividuals`, when applying this pattern to describe connections between them. A `Connection` is established between two `TopologicalIndividuals`. The concept `Connection` represents an *n-ary* relationship

[18]Example use of componency ODP: http://mowl-power.cs.man.ac.uk/2011/07/sssw/lp.html.

[19]https://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl.

[20]TimeIndexedPartOf: http://ontologydesignpatterns.org/wiki/Submissions:TimeIndexedPartOf.

[21]Part-whole modelling in ISO15926: https://www.posccaesar.org/wiki/ISO15926inOWLPart2.

[22]Constituency ODP: http://ontologydesignpatterns.org/wiki/Submissions:Constituency.

between two components: it points to the two components that are involved in the connection, it specifies the direction of the connection and also specifies the component level `Connectors` that are involved (i.e., a connector corresponds to the notion of a port in UML or SysML).

While in this section the main focus was on how to correctly model engineering knowledge in ontologies, an important next step is the population of engineering ontologies with concrete instances (c.f. guideline Step 7 of Noy and McGuinness 2001). This process requires extracting data from engineering artefacts and converting it into Semantic Web formats, i.e., RDF(S) and OWL. Section 5.7 describes how semantic data may be acquired from engineering artefacts stored in legacy data formats such as spreadsheets, XML files and databases.

5.7 Acquisition of Semantic Knowledge from Engineering Artefacts

Section 5.2 describes in detail different strategies for ontology engineering. In short, two main phases can be distinguished in the ontology development process: (1) constructing the ontology skeleton (i.e., classes and properties) that can be achieved either manually or by using *ontology learning techniques* (Maedche 2012); and (2) creating ontology instances, a process known as *ontology population* (Petasis et al. 2011).

Ontology learning and population for engineering applications often require extracting data from legacy systems and existing proprietary data. Considering the scale of data in such systems, the (automated) tool-support for these processes is of vital importance, as performing them manually is time-consuming and error-prone. Engineering data comes in many formats, most commonly as (semi-) structured data, i.e., spreadsheets, XML, and databases (Villazón-Terrazas et al. 2010).

Spreadsheets as data representation format are often used to share, store, and exchange data in engineering environments despite of a set of important drawbacks, such as: the implicit data schema hampers automatic and effective processing; high-level of freedom and, therefore, high variability in data representation, which often does not adhere to best practices of data representation. These weaknesses are balanced by the positive characteristics of spreadsheets. Indeed, from a user point of view, spreadsheets are easy to understand, they do not require sophisticated skills to create and work with; and have adequate representational power and expressiveness for many common tasks.

XML facilitates the encoding of documents readable for both humans and machines. XML stores data in plain text, wrapping the information in tags. XML supports nested elements, which allows easy capturing of hierarchical structures omnipresent in engineering. The fact that XML does not restrict the use to predefined tags has both advantages and disadvantages. On the positive side, it leads to high flexibility in defining XML document structures according to the user or corporate preferences. However, XML (and its extensions) can become overly verbose and complex. An additional drawback is that XML does not define the semantics of data, but rather its structure, leaving space for ambiguity in data interpretation (Bray et al. 2008).

A large amount of engineering data (similar to other domains) is stored *in relational databases* (RDBs) thanks to their maturity of technology and tools, enabling support for scalability, optimized query execution, efficient storage and security (Sahoo et al. 2009). RDBs represent data in structured ways and, if modeled according with the best practices potentially incorporate significant share of domain knowledge, especially in a large company, where RDBs are typically maintained over a long period. This makes RDBs a valuable source for data extraction as not only the stored data, but also the schemas, defined queries and procedures can be used (with the support of domain experts) for ontology learning and population (Spanos et al. 2012).

Besides the three main data formats described above, a variety of other company- or discipline-specific proprietary formats are used in multi-disciplinary engineering settings. In this section we will focus on acquiring semantic knowledge from spreadsheets, XML-based documents, and databases.

The Semantic Web community and tool vendors have developed numerous tools and mapping techniques to enable (semi-)automated data transformation from legacy data sources into Semantic Web formats. For instance, the following tools can be used for extracting data *from spreadsheets* and converting that data into Semantic Web formats: *Open Refine* (former *Google Refine*),[23] *Anzo* for Excel[24] from Cambridge Semantics, RDF123 (Han et al. 2006), *XLWrap* (Langegger and Wöß 2009), *MappingMaster* plug-in for *Protégé* (O'Connor et al. 2010) and the Cellfie plugin,[25] and to some extent *Populous* (Jupp et al. 2012). For *converting from XML* files one can use Protégé's *XMLTab,*[26] *Rhizomik ReDeFer*[27] or the *xSPARQL* language (Bischof et al. 2012). Data *transformation from an RDB* into OWL and RDF formats is supported by e.g., *Relational.OWL* (De Laborda and Conrad 2005), the *DataMaster* plug-in for Protégé (Nyulas et al. 2007), and D2RQ (Bizer and Cyganiak 2006). Also, much work on this topic is covered within the W3C RDB2RDF Working Group[28] which focuses on the development of R2RML,[29] a language to define customized mappings from relational databases to RDF datasets. Some tools support generating semantic data *from various types of data sources*, e.g., the RML language (Dimou et al. 2014) allows specifying

[23]Open Refine: http://openrefine.org.

[24]Anzo for Excel: http://www.cambridgesemantics.com/solutions/spreadsheet-integration.

[25]Cellfie: https://github.com/protegeproject/cellfie-plugin/wiki.

[26]Protégé XML Tab: http://protegewiki.stanford.edu/wiki/XML_Tab.

[27]Rhizomik ReDeFer: http://rhizomik.net/html/redefer/.

[28]W3C's RDB2RDF Working Group: https://www.w3.org/2001/sw/rdb2rdf/.

[29]R2RML: https://www.w3.org/TR/r2rml/.

mappings from CSV and XML to RDF, and TopBraid Composer[30] from TopQuadrant can manage spreadsheets, XML and databases.

Another approach to manage the legacy engineering data is *ontology-based data access (OBDA)*. According to OBDA, an ontology is used as a mediator to access local data sources. Here data remains stored in the original databases and the ontology defines a global schema that provides a common vocabulary for query formulation, thus separating user from the details of the actual structure of the local data sources. The OBDA-based system rewrites user queries (formulated in terms of global schema) into queries built in terms of local data sources, and then delegates the query execution to data sources (Rodriguez-Muro et al. 2008; Civili et al. 2013). This approach is especially intended for applications that rely on large amounts of data. An example of tool supporting the ODBA is Ontop (Rodriguez-Muro et al. 2013).

As the available tools for data extraction and transformation from legacy data formats vary in many aspects (e.g., input and output formats supported, license and price, or a required level of user expertise) it can be difficult to select the best fitting tool for a specific usage context. Therefore, practitioners need to be supported to select the most suitable tool for their context.

To address the need for supporting practitioners in choosing the most appropriate data transformation tool for their context, Kovalenko et al. (2013) developed a *tool selection framework* as a means to facilitate choosing appropriate tools for ontology population from spreadsheet data. The framework has been applied to a set of transformation tools from spreadsheets. The developed framework considers nine criteria: *general information* (maturity, license and type of tool-plug-in or stand-alone tool); *usability* (availability of GUI and required user knowledge to start working with a tool); supported *input/output formats*; *mapping definition* aspects (i.e., how mappings are represented internally and to the user; and how they are stored); *expressiveness* (what complexity of data can be managed with a tool); *multi-user support*; *required additional software* (e.g., for plug-ins); and *additional features* (any other functionality that is provided by a tool). The proposed framework is used to classify the existing tools, thus, providing a focused summary on selection-critical aspects of tools for non-Semantic Web experts. Depending on a project and available project resources (e.g., budget or availability of knowledge engineer), various weights will be assigned for different criteria. The selection framework is adaptable based on specific engineering project requirements and needs. Similar tool selections frameworks should be developed to facilitate the selection of ontology learning and population tools from XML documents and relational databases for engineering practitioners.

[30]TopBraid Composer: http://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/.

5.8 Summary and Future Work

This chapter aimed to provide introductory material on major issues related to the semantic modelling and acquisition of engineering knowledge that potential adopters of SWTs should consider. Concretely, the chapter introduced and discussed the following key topics:

- *Ontology engineering methodologies* prescribe concrete steps for creating and maintaining ontologies (Sect. 5.2). A wealth of methodologies exists focusing on different settings and contexts, starting from generic ontology creation guidelines suitable for beginners (the guidelines of Noy and McGuinness), to methodologies that cover a broader view of ontology engineering including their use in applications and organizations (On-to-Knowledge, METHONTOLOGY), scenario-based methodologies (NeOn Methodology) as well as methodologies suitable in collaborative ontology engineering settings (DILLIGENT).
- *Ontology evaluation* is an important issue that needs to be considered when building, reusing or modularizing ontologies (Sect. 5.3). There is an abundance of techniques and metrics to perform ontology evaluation, but their choice depends on the goal of the evaluation, which, most often focuses on one of the following issues: domain coverage, quality of modelling, suitability for an application or task and adoption by the community.
- *Classification of engineering ontologies* is currently an open research topic, with the exception of an initial classification scheme, which was derived based on a review of the content of existing ontologies by Legat et al. (2014). In Sect. 5.4, we report on aligning this classification scheme with the *PPR* view in order to create a classification scheme that is meaningful to ontology experts and industry practitioners alike.
- *Examples of engineering ontologies* in Sect. 5.5 give an insight into the variety of the existing ontologies and demonstrate the usefulness of the previously proposed scheme for ontology classification.
- *Ontology modelling with ontology patterns described in* Sect. 5.6, approaches the issue of semantic modelling in more depth. It provides examples of recurring, engineering-specific modelling needs, and shows how these can be addressed, for example, by modelling best practices, such as ontology design patterns.
- *Ontology learning and population from legacy data formats* are important tasks as they facilitate the transition from legacy information systems to Semantic Web-based solutions. As discussed in Sect. 5.7, engineering data is most often stored in databases, XML files or spreadsheets. Several tools are available for transforming data from these formats into ontologies and ontology instances. A major issue for practitioners is the selection of the most appropriate tool for their context. This issue is alleviated through tool selection frameworks, such as the one developed for evaluating and selecting tools for translating spreadsheet data.

The following topics emerged as interesting for future investigations. There is a clear need for supporting both industry adopters and Semantic Web experts with tools for identifying existing ontologies. A prerequisite is the availability of ontology classification schemes, which can be easily understood by both stakeholder groups, as well as the availability of surveys that would provide a comprehensive view of engineering ontologies. We expect that ontology reuse would be highly facilitated, if these elements were in place. For supporting the actual modelling of ontologies ODPs are useful. However, these are currently presented in a domain-agnostic manner, which hampers their adoption. Future work could therefore also investigate how to bring ODPs closer to creators of engineering ontologies. This step could involve, for example, a catalogue of frequently emerging modelling needs and guidelines of solving these with ODPs adapted to the engineering domain. Finally, practitioners would highly benefit from the availability of tool selection frameworks that support them in evaluating and selecting the most suitable tools for their context of data transformation from legacy data sources.

Acknowledgments This work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth, and the National Foundation for Research, Technology and Development in Austria.

References

Alani, H., Brewster, C., Shadbolt, N.: Ranking ontologies with AKTiveRank. In: 5th International Semantic Web Conference (ISWC), Athens, GA, USA, pp. 1–15 (2006)

Alsafi, Y., Vyatkin, V.: Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. J. Robot. Comput. Integr. Manuf. **26**(4), 381–391 (2010)

d'Aquin, M., Noy, N.F.: Where to publish and find ontologies? A survey of ontology libraries. J. Web Semant. **11**, 96–111 (2012)

Artale, A., Franconi, E., Guarino, N., Pazzi, L.: Part-whole relations in object-centered systems: an overview. Data Knowl. Eng. **20**(3), 347–383 (1996)

Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. J. Data Semant. **1**(3), 147–185 (2012)

Bizer, C., Cyganiak, R.: D2R server-publishing relational databases on the semantic web. In: Poster at the 5th International Semantic Web Conference, pp. 294–309 (2006)

Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible markup language (XML) 1.0 (2008)

Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data driven ontology evaluation. In: 4th International Conference on Language Resources and Evaluation (LREC), Lisbon, Portugal, pp. 164–169 (2004)

Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng. 84–102 (2005)

Blázquez, M., Fernández-López, M., García-Pinar, J.M., Gómez-Pérez, A.: Building ontologies at the knowledge level using the ontology design environment. In: Gaines, B.R., Musen, M.A. (eds.) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW), Banff, Canada, SHARE4:1–15 (1998)

Cantador, I., Fernandez, M., Castells, P.: Improving ontology recommendation and reuse in WebCORE by collaborative assessments. In: Workshop on Social and Collaborative Construction of Structured Knowledge, 16th International World Wide Web Conference (WWW) (2007)

Civili, C., Console, M., De Giacomo, G., Lembo, D., Lenzerini, M., Lepore, L., Mancini, R., et al.: MASTRO STUDIO: managing ontology-based data access applications. Proc. VLDB Endow. **6**(12), 1314–1317 (2013)

Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., et al.: The SSN ontology of the W3C semantic sensor network incubator group. J. Web Semant. **17**, 25–32 (2012)

Corcho, O., Fernández-López, M., Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies: Where is their meeting point? Data Knowl. Eng. **46**(1), 41–64 (2003)

De Laborda, C.P., Conrad, S.: Relational.OWL: a data and schema representation format based on OWL. In: Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling, vol. 43, pp. 89–96. Australian Computer Society (2005)

Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Proceedings of the 7th Workshop on Linked Data on the Web (LDOW) (2014)

Drath, R. (ed.): Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA. Springer DE (2010)

Fernandez, M., Cantador, I., Castells, P.: CORE: a tool for collaborative ontology reuse and evaluation. In: 4th International Workshop on Evaluation of Ontologies for the Web at the 15th International World Wide Web Conference (WWW 2006), Edinburgh, Scotland (2006)

Fernandez, M., Overbeeke, C., Sabou, M., Motta, E.: What makes a good ontology? A case-study in fine-grained knowledge reuse. In: 4th Asian Semantic Web Conference (ASWC), Shanghai, China, pp. 61–75 (2009)

Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: from ontological art towards ontological engineering. In: Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, pp. 33–40 (1997)

Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S. et al. (eds.) Handbook of Ontologies, 2nd edn., pp. 221–244. Springer (2009)

Gomez-Perez, A., Corcho, O., Fernandez-Lopez, M.: Ontological Engineering: With Examples from the Areas of Knowledge Management, 404 p. Springer (2004)

Guarino, N., Welty, C.: An overview of OntoClean. In: Handbook on Ontologies, pp. 151–172. Springer, Berlin (2004)

Han, L., Finin, T., Parr, C., Sachs, J., Anupam, J.: RDF123: a mechanism to transform spreadsheets to RDF. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI). AAAI Press (2006)

Hepp, M.: Products and services ontologies: a methodology for deriving OWL ontologies from industrial categorization standards. Int. J. Semant. Web Inf. Syst. (IJSWIS) **2**(1), 72–99 (2006)

IEC 62424: Representation of process control engineering—Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools (2008)

IEC 62714 (all parts): Engineering data exchange format for use in industrial systems engineering —Automation Markup Language

Industrial automation systems and integration—Integration of life-cycle data for process plants including oil and gas production facilities. http://www.iso.org/

Janowicz, K., Compton, M.: The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In: Taylor, K., Ayyagari, A., Roure, D. (eds.) The 3rd International Workshop on Semantic Sensor Networks (SSN10) at the 9th International Semantic Web Conference (ISWC) (2010)

Jupp, S., Horridge, M., Iannone, L., Klein, J., Owen, S., Schanstra, J., Wolstencroft, K., Stevens, R.: Populous: a tool for building OWL ontologies from templates. BMC Bioinform. **13**(1) (2012)

Kovalenko, O., Serral, E., Biffl, S.: Towards evaluation and comparison of tools for ontology population from spreadsheet data. In: Proceedings of the 9th International Conference on Semantic Systems, pp. 57–64. ACM (2013)

Langegger, A., Wöß, W.: XLWrap—Querying and Integrating Arbitrary Spreadsheets with SPARQL. Springer, Berlin (2009)

Legat, C., Seitz, C., Lamparter, S., Feldmann, S.: Semantics to the shop floor: towards ontology modularization and reuse in the automation domain. In: 19th IFAC World Congress (2014)

Lenat, D.B., Guha, R.V.: Building Large Knowledge-Based Systems: Representation and Inference in the CycProject. Addison-Wesley, Boston (1990)

Lohse, N., Hirani, H., Ratchev, S.: Equipment ontology for modular reconfigurable assembly systems. Int. J. Flex. Manuf. Syst. **17**(4), 301–314 (2006)

Maedche, M., Staab, S.: Measuring similarity between ontologies. In: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW), pp. 251–263 (2002)

Maedche, A.: Ontology Learning for the Semantic Web, vol. 665. Springer Science & Business Media (2012)

Morbach, J., Wiesner, A., Marquardt, W.: OntoCAPE—A (re)usable ontology for computer-aided process engineering. Comput. Chem. Eng. **33**(10), 1546–1556 (2009)

Novák, P., Serral, E., Mordinyi, R., Šindelář, R.: Integrating heterogeneous engineering knowledge and tools for efficient industrial simulation model support. Adv. Eng. Inform. **29**, 575–590 (2015)

Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology, Stanford University Knowledge Systems Laboratory Technical Report KSL-01-05 (2001)

Nyulas, C., O'Connor, M., Tu, S.: DataMaster—a plug-in for importing schemas and data from relational databases into Protege. In: Proceedings of the 10th International Protege Conference (2007)

O'Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping Master: a flexible approach for mapping spreadsheets to OWL. In: The Semantic Web–ISWC, pp. 194–208. Springer, Berlin (2010)

Odell, J.J.: Six different kinds of composition. J. Object Oriented Program. **5**(8), 10–15 (1994)

Object Management Group (OMG): OMG Systems Modeling Language Specification. http://www.sysml.org/docs/specs/OMGSysML-FAS-06-05-04.pdf (2006)

Petasis, G., Karkaletsis, V., Paliouras, G., Krithara, A., Zavitsanos, E.: Ontology population and enrichment: state of the art. In: Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, pp. 134–166. Springer (2011)

Pinto, H.S., Tempich, C., Staab, S.: DILIGENT: towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), pp. 393–397. IOS Press (2004)

Porzel, R., Malaka, R.: A task-based approach for ontology evaluation. In: Proceeding of the ECAI Workshop on Ontology Learning and Population (2004)

Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy Pitfall Scanner!): an on-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**(2) (2014)

Poveda-Villalón, M.P.: Ontology Evaluation: a pitfall-based approach to ontology diagnosis. Ph.D. Thesis, UPM Madrid (2016)

Rector, A., Welty, C.: Simple part-whole relations in OWL Ontologies. W3C Editor's Draft 11 (2005)

Rodriguez-Muro, M., Lubyte, L., Calvanese, D.: Realizing ontology based data access: a plug-in for Protégé. In: IEEE 24th International Conference on Data Engineering Workshop (ICDEW), pp. 286–289. IEEE (2008)

Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: ontop of databases. In: The Semantic Web, ISWC, pp. 558–573. Springer, Berlin (2013)

Sabou, M., Fernandez, M.: Ontology (network) evaluation. In: Suarez-Figueroa, M.C., et al. (eds.) Ontology Engineering in a Networked World. Springer, Berlin (2012)

Sabou, M., Ekaputra, F.J., Kovalenko, O.: Supporting the engineering of cyber-physical production systems with the AutomationML analyzer. In: Proceedings of the CPPS Workshop, at the Cyber-Physical Systems Week. Vienna (2016)

Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to RDF. W3C RDB2RDF Incubator Group Report (2009)

Schleipen, M., Drath, R.: Three-View-Concept for modeling process or manufacturing plants with AutomationML. In: IEEE Conference on Emerging Technologies & Factory Automation (ETFA) (2009)

Sicilia, M., Garcia-Barriocanal, E., Sanchez-Alonso, S., Rodriguez-Garcia, D.: Ontologies of engineering knowledge: general structure and the case of software engineering. Knowl. Eng. Rev. **24**(3), 309–326 (2009)

Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: a survey. Semant. Web J. **3**(2), 169–209 (2012)

Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge processes and ontologies. IEEE Intell. Syst. **16**(1), 26–34 (2001)

Stevens, R., Brook, P., Jackson, K., Arnold, S.: Systems Engineering: Coping with Complexity. Prentice Hall PTR (1998)

Suárez-Figueroa, M.C.: NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. Dissertations in Artificial Intelligence, vol. 338. IOS Press (2012)

Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The NeOn methodology framework: a scenario-based methodology for ontology development. Appl. Ontol. **10**(2), 107–145 (2015)

Verein Deutscher Ingenieure (VDI): Design methodology for mechatronic systems. VDI-Richtlinie 2206. Beuth Verlag, Berlin (2004)

Villazón-Terrazas, B., Suárez-Figueroa, M.C., Gómez-Pérez, A.: A pattern-based method for re-engineering non-ontological resources into ontologies. Int. J. Semant. Web Inf. Syst. (IJSWIS) **6**(4), 27–63 (2010)

Winston, M., Chaffin, R., Hermann, D.: A taxonomy of part-whole relations. Cogn. Sci. **11**(4), 417–444 (1987)