

Ontology for Industrial Engineering: A DOLCE Compliant Approach

Walter Terkaj¹, Stefano Borgo² and Emilio M. Sanfilippo²

¹*Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato (STIIMA-CNR) Via A. Corti, 12, 20133, Milano, Italy*

²*ISTC-CNR Laboratory for Applied Ontology, via alla cascata 56/C, 38123, Povo, Trento, Italy*

Abstract

Recent academic and industrial initiatives show an increasing interest in the use of foundational ontologies to support the development of domain-specific ontologies. Foundational ontologies guarantee formal and conceptual robustness, and at the same time support the integration of multiple domain-ontologies aligned to the same foundational ontology. This paper reports some ongoing work in this area focusing on the application of the DOLCE ontology in the manufacturing domain. Taking advantage of the newly formal representation of DOLCE in OWL and building on previous modeling works, the paper shows how different modeling strategies can be used depending on the engineering requirements at hand, and discusses advantages and disadvantages of these strategies. We exemplify the discussion by means of an industrial case study.

Keywords

Ontology, Foundational Ontologies, DOLCE, Industrial engineering

1. Introduction

Recent initiatives like the European project OntoCommons¹ and activities in OAGI like the Industrial Ontologies Foundry (IOF)² show that there is a convergence in scientific and industrial interests towards the development of ontology-based information systems for industry at large.

While this convergence can be analyzed from different perspectives, the modelling of industrial engineering application cases asks for a common and extensible data model for the representation of assets related to systems, resources, processes, and products. This can be done from different perspectives (e.g., relative to one stakeholder or another) and at different levels of granularity (depending, e.g., on the information available or on contextual constraints).


To considering these industrial models for implementation purposes, the ontological model must be provided in some computational language that can be directly used at the organization's site, independently whether the model is for production, logistics or management. The adopted computational language may change considerably depending on several issues like legacy constraints and local expertise but, by and large, one usually finds applications based on

FOMI 2022: 12th International Workshop on Formal Ontologies meet Industry, September 12–15, 2022, Tarbes, France

✉ walter.terkaj@stiima.cnr.it (W. Terkaj); stefano.borgo@cnr.it (S. Borgo); emilio.sanfilippo@cnr.it (E. M. Sanfilippo)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://ontocommons.eu/>

²<https://www.industrialontologies.org/>

languages that have been developed within the Semantic Web approach. Today, the default choice seems to be to work with some version of the OWL language³ due to its logical properties: computability, decidability and, depending on needs and constructs, efficiency. Efficiency is particularly relevant because there are different kinds of reasoning tasks that may be performed at runtime. They are classified and studied in terms of data complexity, taxonomic complexity, query complexity, and combined complexity.⁴

While OWL is usually considered a suitable solution for ontology implementations, the adoption of OWL remains the result of a compromise. On the one hand, OWL allows to address decision making needs that occur at runtime. On the other hand, it is strong enough to model the taxonomic structure of the ontology and some aspects of its relational structure. Nonetheless, top-level or middle level ontologies, which are at the core of robust ontology modeling, are preferably represented using stronger logical languages like first-order logic (FOL) [1]. Indeed, there is a gap between the information an ontology can model in FOL and the information the OWL language can represent. This gap has important consequences since there exist different approaches to generate an OWL version of an ontology which is initially given in FOL [2]. For instance, given that not everything can be expressed in OWL, it is unclear which parts of the FOL ontology should be given priority when generating the OWL version. Other problems arise when considering interoperability across ontologies as it is unclear today which type of OWL version of a domain ontology optimises the interoperability alignments that are under development across top-level and/or middle-level ontologies (an ongoing effort taken by the OntoCommons project among others). This paper focuses on the first of these issues aiming, as we will see later, to show what the difficulties are and some options that should be considered. Note that the existence of an ontology in two different languages is not *per se* a problem, since an ontology is not a syntactic entity, but a semantic one. Two versions of the same ontology may coexist as long as they share the same class of models.⁵ However, when the two languages have different expressive power, like FOL and OWL do, the classes of models are different. Practically speaking, the two ontologies describe different views of reality, and derive different consequences even when describing the same scenario.

This problem is not restricted to some ontological systems, it applies to all ontologies written in such languages. Actually, the problem is even more general, since it holds for logical theories at large, but it becomes particularly relevant when ontologies are at stake due to their general aim to provide a robust understanding of (a fragment of) reality. Furthermore, the problem cannot be classified as a logical fact that we simply have to live with, as there are different ways to approximate the richer ontology in a less expressive language. This means that there are choices one can (and perhaps needs to) make. Given that there are different approximation options, one wonders if there is the possibility to develop guidelines or even a methodology to minimize the modeling gap.

The aim of this paper is to highlight the aforementioned problem and see what options are generally available. We take the perspective of a practitioner and, thus, exemplify the discussion (which remains general) on a specific system, namely, the DOLCE ontology [3]. DOLCE is under

³<https://www.w3.org/TR/owl-overview/>

⁴https://www.w3.org/TR/owl-profiles/#Computational_Properties

⁵Things are more complex than this, as different languages have different ontological commitments. Generally speaking, this is an important point in applied ontology but, to keep things simple, we ignore it in this paper.

evaluation as part 3 of the ISO standard 21383⁶ which includes two versions of the ontology, one in Common Logic Interchange Format⁷ (CLIF), and one in OWL (see Section 3). This means that both the FOL and the OWL versions are official versions of DOLCE.⁸

The paper is structured as follows. Section 2 introduces the requirements of a domain ontology for manufacturing applications, whereas Section 3 presents the proposal of an OWL ontology extending DOLCE for such domains. Examples of instantiation are discussed in Section 4 and conclusions are drawn in Section 5.

2. Problem Statement

Industrial systems are typically characterized by a long lifecycle, including phases like early design, detailed design, installation and ramp-up, run and monitor, maintenance, and decommissioning. Therefore, a data model for industrial systems must be able to cope with evolving data that refer to the same concept but with a different status and level of details (LOD). Indeed, all planning activities are particularly data-intensive. For instance it is necessary to deal with capacity plans, process plans, and production plans. The first addresses the timed acquisition of production resources, e.g., machine tools, transporters, fixtures, etc., to meet production goals. A process plan gives work instructions to transform a part from its initial form to the final form and may include the description of manufacturing processes, operational setup, process parameters, equipment and/or machine tool selection. Finally, a production plan determines the production resources assigned to execute operations on each workpiece, the scheduling and resource optimization [5]. As anticipated in the introduction, the choice of the approach may be pragmatic and related to local requirements and modeling needs. In principle, an ontology should allow these modeling views to coherently coexist and interact.

Since also the relationships between assets evolve over time, it may be necessary to add the time dimension to capture changing configurations and plans. However, this may significantly increase the complexity of an ontology, in particular if modeled in OWL.

Narrowing the focus on manufacturing engineering, listing some relevant competency questions can help to highlight the importance of modeling relations between assets:

CQ1 Which are the descriptions of artefacts (i.e., part types) to be produced by a production system under design?

CQ2 Which are the components of an artefact as designed?

CQ3 Which are the components of a physical artefact?

CQ4 Which are the executable activities (i.e., process plans) to obtain an artefact as output?

⁶<https://www.iso.org/obp/ui#!iso:std:78927:en>

⁷For the CLIF standard see: <https://www.iso.org/standard/66249.html>

For DOLCE in CLIF see: <http://www.loa.istc.cnr.it/wp-content/uploads/2021/07/dolce-mace4-prover9.zip>

⁸This year marks the twentieth anniversary of the official version of DOLCE [4] which is based on first-order modal logic. The new versions in CLIF and OWL that we are mentioned, have been generated to comply with the requirements of the new ISO standard 21383. As of now these versions should be considered in a developing phase due, among other things, to the alignment problems we are in part addressing in this paper.

- CQ5** Which are the sub-activities nesting a complex activity (e.g., a process plan)?
- CQ6** Which are the activity occurrences nesting a complex activity occurrence (e.g., a manufacturing operation)?
- CQ7** Which is the artefact description (activity) that an artefact (activity occurrence) is implementing?
- CQ8** Which production resource is needed to execute an activity?
- CQ9** Which production resource is scheduled to execute an activity occurrence (e.g., a manufacturing operation)?
- CQ10** Which are the capabilities of a machine tool in terms of activity occurrence that is able to execute?

These questions address the variety of aspects that the ontology should manage to cover. The long-term goal is to obtain a recommended extension of an OWL ontology (e.g., an OWL version of DOLCE) suitable for industrial engineering applications.

3. DOLCE for Industrial Engineering

The OWL DOLCE version (prefix *dolce* in Table 1) is here tested as the core module for possible extensions to address the area of industrial engineering. This version is named DolceEng (see prefix *d4e* in Table 1).

The proposed extension takes advantage of previous contributions in this domain, in particular:

- The standard ifcOWL ontology [6] (prefix *ifc* in Table 1) that is the official OWL version of the Industry Foundation Classes (IFC) standard [7]. IFC provides a wide range of high-level and detailed concepts to represent systems in Architecture, Engineering, and Construction (AEC). However, the OWL version of the standard is hindered by a lack of clear ontological commitment [8] and by an extremely complex monolithic structure [9]. In addition, ifcOWL relies on reified relations [8] resulting in verbose and computationally expensive instantiations.
- A domain ontology for digital factory applications named Factory Data Model (FDM). This data model,⁹ already presented in previous works [10, 11], has been developed as an OWL ontology to represent factory assets related to production systems, resources, processes, and products. The ontology integrates different knowledge domains while reusing existing technical standards (e.g., ifcOWL [6], W3C SSN/SOSA [12], UML Statechart [13]). In particular, an extension to ifcOWL has been adopted (prefix *ifcext* in Table 1) to simplify the representation of relations between classes, similarly to what proposed by [14].

The design of DolceEng is presented in the following subsections focusing on taxonomy (Sect.3.1) and relations (Sect.3.2), while recalling the competency questions listed in Sect.2.

⁹<https://virtualfactory.gitbook.io/vlft/kb/fdm>

Table 1

Prefixes of ontology modules

Prefix	Prefix IRI of ontology module
dolce	http://www.loa.istc.cnr.it/dolce/dolce-owl/DOLCE#
d4e	http://www.ontoeng.com/DolceEng#
ifc	https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD1/OWL#
ifcext	http://www.ontoeng.com/IFC4_ADD1_extension#

3.1. Taxonomy

The taxonomy of DolceEng specializes OWL classes defined in DOLCE by simple subsumption relations (*rdfs:subClassOf*). The relevant classes are reported in Table 2 and have been introduced in [5]. From a general perspective, the table covers perdurants (aka events, occurrences), i.e., happenings in time like *d4e:ActivityOccurrence*, endurants like physical objects (e.g., *d4e:TechnicalArtefact*), qualities (e.g., *d4e:Capability*), concepts (e.g., *d4e:Role*), and descriptions (e.g., *d4e:ArtefactDescription*, *d4e:Activity*). These latter two are useful for different purposes; e.g., in engineering design or manufacturing planning scenarios, experts need to explicitly represent the design models (or process plans) with which technical artefacts (or activity occurrences) are meant to comply (see **CQ1**). The *d4e:Description* class can be used for these purposes thanks to specific relations (see next subsection 3.2). In research work about DOLCE, the class of concepts (*d4e:Concept*) has been introduced to model the social or cognitive nature of certain properties [15]. For instance, *d4e:Concept* can be used to capture properties that entities satisfy within specific manufacturing contexts. Roles themselves (*d4e:Role*) are examples of concepts, for instance, *being mechanical engineer*, *employee*, and *lathe turner*. In addition, following the analysis by Sanfilippo et al. [5], roles can also model manufacturing resources, i.e., physical entities which satisfy the requirement(s) required for the execution of a manufacturing activity (plan). For instance, an individual lathe machine has the role of manufacturing resource when that machine satisfies the requirements of a manufacturing activity, e.g., to have the capability of cutting certain metallic materials within specific tolerance ranges.

DolceEng, as shown in Table 2, covers also classes that do not belong to DOLCE, like capabilities and capacities (see **CQ10**). We resume here the view that capabilities are DOLCE individual qualities associated with functions or activities, as discussed in [16]. This means to take capabilities as relational qualities that depend on the environment and functionality, so that it is possible to model the meaning of expressions like ‘it has the capability to hold’ (said of a container), ‘it has the capability to carry’ (said of a truck), ‘it has the capability to cut’ (said of a blade) [16]. Taking capabilities as a kind of contextual quality (the capability of holding is implicitly parameterised by the qualities of the objects to be held like size, weight and material), one is driven to separate intrinsic qualities (like shape, weight and color) and relational qualities (like the capability to hold, to carry, to acquire information and so on). In engineering, the *designed capability* acquires a special position. These are capabilities that are attributed (by the designer and/or the producer) to the designed/produced object.

Borgo et al. [16] contrasted capabilities and capacities proposing to see the latter as qualities that provide quantitative information. In this view, the capacities of an object are the qualities that specify to which extent the object can realize a function or, in other terms, can manifest

a capability. This view classifies capacities as a type of relational qualities that, while distinct from capabilities, also depend on the context in which the object is used. Examples are the containment capacity (essentially the volume available to perform a storage functionality and which depends on the type of entity one aims to store, e.g., liquid vs solid items of a certain form and size) and the throughput (e.g., the number of products produced in a certain time period by a production system with the capability to produce that kind of products).

Table 2

Classes defined in DolceEng

owl:Class	rdfs:subClassOf
d4e:ActivityOccurrence	dolce:Accomplishment
d4e:Artefact	dolce:NonAgentivePhysicalObject
d4e:TechnicalArtefact	d4e:Artefact
d4e:Building	d4e:TechnicalArtefact
d4e:TransformationSystem	d4e:TechnicalArtefact
d4e:Concept	dolce:NonAgentiveSocialObject
d4e:Role	d4e:Concept
d4e:Description	dolce:NonAgentiveSocialObject
d4e:Activity	d4e:Description
d4e:ArtefactDescription	d4e:Description
d4e:TechnicalArtefactDescription	d4e:ArtefactDescription
d4e:Capability	dolce:PhysicalQuality
d4e:Capacity	dolce:PhysicalQuality

3.2. Relations

Relations are key elements in conceptual models, ontologies included, to link classes and establish formal restrictions. Table 3 compares how some relevant relations needed to represent industrial systems, in particular manufacturing systems (cf. competency questions in Section 2), can be modeled with 1) the standard *ifcOWL* ontology using reified relations (*owl:Class*), 2) the Factory Data Model (FDM) using the *ifcOWL* extension *ifcext* that employs binary relations (*owl:ObjectProperty*), and 3) *DOLCE* as employed by the proposed extension *DolceEng* taking advantage of both binary and reified relations.

From a modeling perspective, there is an important distinction in using either temporalized or non-temporalized relations. For instance, one may simply say that a physical lathe has a cutting tool as component, with or without explicitly stating the time when the relation holds. In a certain interpretation¹⁰ one aspect that changes from a formal perspective is the arity of the relation. Without reference to time, one can use a binary logical predicate like *hasComponent(x, y)*, whereas a ternary predicate is needed to include also the time *t* at which *x* is a component of *y*, i.e., *hasComponent(x, y, t)*. Adopting one view or the other one has consequences in the expressivity of the application models. For instance, the mereological characterization of physical entities commonly includes time variables, because physical objects can undergo changes in time with respect to their parts [18]. Hence, an ontology restricted to non-temporalized parthood relations cannot distinguish cases where, e.g., the same tool is

¹⁰See Galton [17], for an overview of different approaches for the representation of time in logical theories.

Table 3Modeling relations in *ifcOWL*, *FDM*, and *DolceEng*

Description of Relation	ifcOWL (owl:Class)	FDM (owl:ObjectProperty)	DolceEng
Decomposition of Artefact Description (hierarchy)	ifc:IfcRelAggregates	ifcext:decomposesObject	dolce:constantProperPartOf
Decomposition of Artefact (hierarchy)	ifc:IfcRelAggregates	ifcext:decomposesObject	dolce:constantProperPartOf, dolce:TemporaryProperPart (reified)
Nesting of Activity	ifc:IfcRelNests	ifcext:nestsObject	dolce:constantProperPartOf
Nesting of Activity Occurrence	ifc:IfcRelNests	ifcext:nestsObject	dolce:temporalProperPartOf
Typing/Modeling definition between Artefact Description and Artefact, Activity and Activity Occurrence	ifc:IfcRelDefinesByType	ifcext:typesObject	d4e:constantlyClassifies, d4e:Classification (reified)
Product Description defined as input of an Activity	ifc:IfcRelAssignsToProcess	ifcext:hasAssignedObject	d4e:hasInputReq
Product Description defined as output of an Activity	ifc:IfcRelAssignsToProduct	ifcext:hasAssignedObject	d4e:hasOutputReq
Assignment of a Resource to an Activity	ifc:IfcRelAssignsToProcess	ifcext:hasAssignedObject	d4e:hasResourceReq
Assignment of a Resource to an Activity Occurrence	ifc:IfcRelAssignsToProcess	ifcext:hasAssignedObject	dolce:Participation (reified)

used at different times with different machines. However, Semantic Web languages, like OWL, force the use of binary relations represented by means of *object* or *data properties*. A trade-off between expressivity and computational tractability is therefore required.

What is interesting from a modeling perspective is that the choice of adopting either a diachronic or synchronic approach depends on the perspective one wishes to represent. Taking as example a product design setting, one may wish to say that the product under design (i.e., a technical artefact description in our terminology) comprises various components without reference to the time when the relation between the whole product and its components holds (see **CQ2**). This is common in bill of materials (BOM) specifications where the relationships between the components of a product are not always temporalized. On the other hand, when representing an assembled product, one may need to refer to the time when an item is attached to another one to enable an explicit representation of change (see **CQ3**). In a nutshell, an ontology for industry should provide users with the flexibility of using either temporalized or non-temporalized relations to meet different requirements in different application scenarios.

To make sense of these considerations, the OWL version of DOLCE (which, as said, is going through a validation period) covers different parthood relations among which three of them are particularly relevant for our purposes: 1) *dolce:TemporaryProperPart* (reified), 2) *dolce:constantProperPartOf*, and 3) *dolce:temporalProperPartOf*. Starting from the first one, *dolce:TemporaryProperPart* stands for the reified ternary relation by which an enduring x is

proper part of y at time t (with x and y being different individuals).¹¹ Being reified, the relation is represented as a class, whose instances are linked to three entities standing for the arguments of the ternary relation. Axioms (Ax1)-(Ax2), in the notation of first-order logic (FOL), say that the first ($tP1$) and second ($tP2$) arguments of the reified relation x are endurants; by (Ax3), the third argument (tPt) is a time. Each instantiation of *TemporaryProperPart* has only these three arguments.

Ax1 $TemporaryProperPart(x) \rightarrow \exists y(tP1(x, y) \wedge Endurant(y))$

Ax2 $TemporaryProperPart(x) \rightarrow \exists y(tP2(x, y) \wedge Endurant(y))$

Ax3 $TemporaryProperPart(x) \rightarrow \exists t(tPt(x, t) \wedge Time(t))$

The second relation, i.e., *dolce:constantProperPartOf* is binary and holds between endurants. It has the intended interpretation that x is proper part of y during the whole existence of y . Accordingly, x can exist before y but once it is related to y via constant proper part, it must remain part of y as long as the latter exists. In the case of physical artefacts, *dolce:constantProperPartOf* can be used to model parthood relations between artefacts and their unreplaceable components, but also for application settings where one does not wish to capture diachronic features of domain entities (see CQ3). This relation can be also used to model the structure of descriptions (recall that the latter are non-physical endurants according to DOLCE) on the assumption that descriptions cannot change parts in time; e.g., a complex activity (i.e., a manufacturing plan) comprising various subactivities or the description of the artefact to be manufactured (see CQ5).¹² The third, binary, relation, *dolce:temporalProperPartOf* is used for parthood among perdurants only; the idea is that a perdurant is an happening. Hence, it is not necessary to parametrize parthood with respect to time [18] (see CQ6).

From Table 3, *d4e:hasInputReq*, *d4e:hasOutputReq*, and *d4e:hasResourceReq* are subrelations of *d4e:hasReq* (has requirement) which is subsumed by (the inverse of) *d4e:constantProperPartOf*. These are used to relate an activity to other descriptions representing what is needed for its executions to take place. In particular, input requirements (*d4e:hasInputReq*) are the entities undergoing an occurrence, e.g., amounts of matter or physical objects; output requirements (*d4e:hasOutputReq*) are the intended results of an activity occurrence (see CQ4); resource requirements (*d4e:hasResourceReq*) capture the resources needed according to the plan (see CQ8), including things like oil, fuel, jigs, etc. (see the next section for an example).

The relation of *participation* is the most general one holding between endurants and the perdurants in which they take part; e.g., the relation between a lathe machine, a turning activity occurrence, and the time at which the former participates in the latter (see CQ9). Similarly to the case of temporary parthood, axioms (Ax4)-(Ax6) represent *Participation* as a class related to three entities standing for the arguments of the reified (ternary) relation: an endurant ($tPC1$), a perdurant ($tPC2$), and time ($tPCt$).

Ax4 $Participation(x) \rightarrow \exists y(tPC1(x, y) \wedge Endurant(y))$

¹¹Endurant in DOLCE is the most general class for things extended in space, e.g., machines, tools, persons, features like holes and bumps. The reification of relationships with arity higher than two is common in Semantic Web approaches, see, e.g., <https://www.w3.org/TR/swbp-n-aryRelations/>.

¹²Following Sanfilippo et al. [5], a description in our ontology can have only other descriptions as parts.

Ax5 $Participation(x) \rightarrow \exists y(tPC2(x, y) \wedge Perdurant(y))$

Ax6 $Participation(x) \rightarrow \exists t(tPCt(x, t) \wedge Time(t))$

The relations for input- and output-participation in (Ax7)–(Ax8) are the counterparts for input and output-requirements, respectively; hence they are needed to qualify the physical entities (satisfying the requirements) taking part in activity occurrences. The participation of resources is not further characterized in formal terms. Specific relations could be however included in DolceEng to distinguish between different resource roles and their participation in activity occurrences.

Ax7 $InputParticipation(x) \rightarrow Participation(x) \wedge \exists yzt(tPC1(x, y) \wedge tPC2(x, z) \wedge tPCt(x, t) \wedge beginOf(t, z))$
(*y* participates in *z* at least at the start time *t* of *z*)

Ax8 $OutputParticipation(x) \rightarrow Participation(x) \wedge \exists yzt(tPC1(x, y) \wedge tPC2(x, z) \wedge tPCt(x, t) \wedge endOf(t, z))$
(*y* participates in *z* at least at the finish time *t* of *z*)

Finally, *classification* is the most general relation holding between either a concept or a description and the entity satisfying it at a certain time.¹³ For instance, considering *employee* as a specific kind of role, we say that John is *classified* by *employee* at time *t* meaning that John plays that role at *t*. Another example, but this time for description, says that a lathe machine is classified by the corresponding product description (e.g., the list of attributes one commonly finds in a product catalog) at *t* meaning that it is compliant with the description at *t*; or a manufacturing activity occurrence of drilling is classified by a manufacturing activity when the happening of the former complies with the restrictions established by the latter (e.g., requirements about the use of resources) (see CQ7). It should be clear that classification requires (at least) three arguments (concept or description, classified entity, and time) in different cases, and has to be therefore reified in OWL. The first argument of the relation (*tCF1*) is either a concept or a description (Ax9); the second argument (*tCF2*) is either an endurant or a perdurant (Ax10); the third argument is time (Ax11).

Ax9 $Classification(x) \rightarrow \exists y((Concept(y) \vee Description(y)) \wedge tCF1(x, y))$

Ax10 $Classification(x) \rightarrow \exists y((Endurant(y) \vee Perdurant(y)) \wedge tCF2(x, y))$

Ax11 $Classification(x) \rightarrow \exists t(Time(t) \wedge tCFt(x, t))$

A binary counterpart expresses *constant* classification. In this case, *d4e:constantlyClassifies* says that *x* (concept or description) classifies *y* for the whole life of *y*, i.e., whenever *y* exists, it is always compliant with *x*. Intuitively, this could be used for capturing a sort of full compliance between a manufacturing occurrence and the corresponding activity; clearly, if the former (while it happens) does not match with the requirements of the latter, it is not classified as an occurrence of the activity. This would likely represent only an ideal case, since manufacturing

¹³We slightly depart from the work of Masolo et al. [15] where *classification* holds only between concepts and the classified entities. We add also descriptions among classifying entities to capture the idea that an individual entity like a product can satisfy the conditions established in a certain description.

occurrences may slightly deviate from their corresponding plans. In this sense, one may think about the specifications of conditions that must be necessarily met by an activity occurrence to be classified by a certain activity. The same consideration could apply to objects.

4. An example based on DolceEng

We show in this section an example about the use of some of the notions of DolceEng introduced in the previous section, while taking as reference the use case of an assembly line¹⁴ presented in [19]. The assembly line produces damped cabinet hinges¹⁵ consisting of 11 components (e.g., Wing, WingScrew, Clip, etc., cf. Table 5 in [19]). The process plan of the hinge is nested by 19 sub-activities of different types (e.g., pin insertion, riveting, pick and place, etc.). Each sub-activity is assigned to a workstation (cf. Table 6 in [19]). A rotating table and linear conveyors are used to transport work-in-progress hinges, thus connecting the workstations.

For instance, the first workstation (*PPW1*) executes a pick and place operation (*pckple1*) moving the main body of the hinge (i.e., the *wing*) from a conveyor to the rotating table. The second workstation (*T1*) executes a *tightening* operation assembling a *screw* on the *wing* component.

The assembly line was previously fully instantiated¹⁶ according to the Factory Data Model [10, 11] to define each asset in terms of properties (e.g., placement, capacity, 3D representation) and relations (e.g., assignments, connections). We show here how DolceEng can be exploited to instantiate the same use case. For the sake of shortness, we show only the tightening operation.

At the description level, formulas (f1)–(f4) represent input, output, and resource requirements for the activity of tightening (recall that requirement-relations are constant-part relations).¹⁷ More specifically, the individual activity *tightening* has input requirements *wing* (f1) and *screw* (f2), output requirement *wingWScrew* (f3) (i.e., a work-in-progress *Hinge* with only the *Wing* and *WingScrew* components), and resource requirement *rscdesT1* (f4), i.e. the description of workstation *T1*.

- $$\begin{aligned} \mathbf{f1} \quad & \text{hasInputReq}(\text{tightening}, \text{wing}) \wedge \\ & \text{Activity}(\text{tightening}) \wedge \text{TechnicalArtefactDescription}(\text{wing}) \\ & \quad \text{(input requirement wing)} \\ \mathbf{f2} \quad & \text{hasInputReq}(\text{tightening}, \text{screw}) \wedge \\ & \text{Activity}(\text{tightening}) \wedge \text{TechnicalArtefactDescription}(\text{screw}) \\ & \quad \text{(input requirement screw)} \\ \mathbf{f3} \quad & \text{hasOutputReq}(\text{tightening}, \text{wingWScrew}) \wedge \\ & \text{Activity}(\text{tightening}) \wedge \text{TechnicalArtefactDescription}(\text{wingWScrew}) \\ & \quad \text{(output requirement wingWScrew)} \end{aligned}$$

¹⁴<https://virtualfactory.gitbook.io/vlft/use-cases/assembly-line>

¹⁵<https://virtualfactory.gitbook.io/vlft/use-cases/factory-assets/assembled-product>

¹⁶<https://difactory.github.io/repository/ontology/VFLab.owl>

¹⁷Some of the domain, range specifications for the relations can be deduced. We include them here to facilitate understanding.

f4 $hasResourceReq(tightening, rscdesT1) \wedge$
 $Activity(tightening) \wedge TechnicalArtefactDescription(rscdesT1)$
(resource requirement $rscdesT1$)

Assume to have a specific occurrence of activity *tightening*, namely, *tightening01* (f5), as well as physical realizations of the descriptions above, namely *wing01* of *wing* (f6), *screw01* of *screw* (f7), *wingWScrew01* of *wingWScrew* (f8)¹⁸, and workstation *T1* of *rscdesT1* (f9).

f5 $constantlyClassifies(tightening, tightening01) \wedge$
 $ActivityOccurrence(tightening01)$

f6 $Classification(cf1) \wedge tCF1(cf1, wing) \wedge tCF2(cf1, wing01) \wedge$
 $tCFt(cf1, t1) \wedge TechnicalArtefact(wing01)$
(classification of *wing01*-input)

f7 $Classification(cf2) \wedge tCF1(cf2, screw) \wedge tCF2(cf2, screw01) \wedge$
 $tCFt(cf2, t2) \wedge TechnicalArtefact(screw01)$
(classification of *screw01*-input)

f8 $Classification(cf3) \wedge tCF1(cf3, wingWScrew) \wedge tCF2(cf3, wingWScrew01) \wedge$
 $tCFt(cf3, t3) \wedge TechnicalArtefact(wingWScrew01)$
(classification of *wingWScrew01*-output)

f9 $Classification(cf4) \wedge tCF1(cf4, rscdesT1) \wedge tCF2(cf4, T1) \wedge$
 $tCFt(cf4, t4) \wedge TechnicalArtefact(T1)$
(classification of *T1*-resource)

Finally, the formulas below represent participation relations between the activity occurrence *tightening01* and workstation *T1* (f10), components *wing01* (f11) and *screw01* (f12), and work-in-progress *wingWScrew01* (f13). Looking at the temporal parameters in the formulas, *dur* is meant to represent the whole duration interval of *tightening01*, whereas *tstart* and *tfinish* represent the beginning and end time of *tightening01*, respectively.

f10 $Participation(pc1) \wedge tPC1(pc1, T1) \wedge tPC2(pc1, tightening01) \wedge tPCt(pc1, dur)$
(workstation *T1* participation)

f11 $InputParticipation(pc2) \wedge tPC1(pc2, wing01) \wedge tPC2(pc2, tightening01) \wedge$
 $tPCt(pc2, tstart) \wedge beginOf(tstart, tightening01)$
(*wing01* input-participation)

f12 $InputParticipation(pc3) \wedge tPC1(pc3, screw01) \wedge tPC2(pc3, tightening01) \wedge$
 $tPCt(pc3, tstart) \wedge beginOf(tstart, tightening01)$
(*screw01* input-participation)

f13 $OutputParticipation(pc4) \wedge tPC1(pc4, wingWScrew01) \wedge tPC2(pc4, tightening01) \wedge$
 $tPCt(pc4, tfinish) \wedge endOf(tfinish, tightening01)$
(*wingWScrew01* output-participation)

¹⁸We do not model further here the structure of *wingWScrew01*, which is an assembled product comprising of both *wing01* and *screw01*.

As said, the formulas cover only part of the example. The aim is to show how the proposed ontology can be used to represent data in industry and, in particular, design and management of manufacturing systems. Further work is needed to improve the formal representation to fully exploit the expressive power of the ontology (e.g., by representing explicitly resources' capacities and capabilities) and complete the implementation of the ontology in OWL language to support ontology-based applications that are able to produce and consume instances (i.e., OWL individuals), while referring to a consolidated foundational ontology like DOLCE .

5. Conclusions

The tendency to adopt robust ontological modeling in engineering is confirmed by current initiatives in the industrial domain. In spite of the many advantages of ontological modeling, the exploitation of ontology from the top-level to the domain level brings to light also problematic aspects which are related to the technical properties of logical languages. It is expected that the analysis and comparison of different use cases together with increasing experiences in real scenario modeling will lead to identify best practices and, hopefully, to design shared guidelines and perhaps even methodologies to optimize domain level ontology development in coherence with the corresponding top-level ontologies.

For this to be achieved, a discussion of the problematic issues has to arise and comparisons of viewpoints have to be developed. This paper contributed to this research line by analysing an industrial use case where an extension of the DOLCE ontology in OWL for industrial engineering can be applied. We have seen that while this leads to complicate OWL data models (also because of the use of reified relations), the patterns one should use are very similar across different relationships like those relating a whole and its parts, an event and its subevents, and a class and its instances.

In the future we plan to make a more systematic analysis of these problems and to test the generality of the results in terms of different case-studies by means of ontology-based applications and reusable SPARQL queries¹⁹ and updates²⁰.

Acknowledgement: The work presented in this paper is partially funded by the European project OntoCommons (GA 958371). We thank colleagues at the ISTC-CNR Laboratory for Applied Ontology, as well as Daniele Porello (University of Genoa, Italy), and Laure Vieu (CNRS, France) for their valuable work on the OWL formalization of DOLCE .

References

- [1] J. Barwise, E. J., The language of first-order logic, CSLI, Stanford, California, 1993.
- [2] T. Hahmann, R. W. Powell II, Automatically extracting owl versions of fol ontologies, in: International Semantic Web Conference, Springer, 2021, pp. 252–269.
- [3] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E. M. Sanfilippo,

¹⁹<https://www.w3.org/TR/sparql11-query/>

²⁰<https://www.w3.org/TR/sparql11-update/>

- L. Vieu, Dolce: A descriptive ontology for linguistic and cognitive engineering, *Applied Ontology* (2022) 1–25.
- [4] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, L. Schneider, DOLCE: A descriptive ontology for linguistic and cognitive engineering, *WonderWeb*, D18 (2003).
- [5] E. M. Sanfilippo, W. Terkaj, S. Borgo, Ontological modeling of manufacturing resources, *Applied Ontology* 16 (2021) 87–109.
- [6] P. Pauwels, T. Krijnen, W. Terkaj, J. Beetz, Enhancing the ifcowl ontology with an alternative representation for geometric data, *Automation in Construction* 80 (2017) 77 – 94.
- [7] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, S. Richter, T. Chipman, M. Weise, J. Wix, Industry Foundation Classes IFC4 official release, 2013. Available online: <https://standards.buildingsmart.org/IFC/RELEASE/IFC4/FINAL/HTML>. Last accessed on 23 February 2020.
- [8] S. Borgo, E. M. Sanfilippo, A. Šojić, W. Terkaj, *Ontological Analysis and Engineering Standards: An Initial Study of IFC*, Springer International Publishing, Cham, 2015, pp. 17–43.
- [9] W. Terkaj, P. Pauwels, A method to generate a modular ifcOWL ontology, in: *CEUR Workshop Proceedings*, volume 2050, 2017.
- [10] W. Terkaj, P. Gaboardi, C. Trevisan, T. Tolio, M. Urgo, A digital factory platform for the design of roll shop plants, *CIRP Journal of Manufacturing Science and Technology* 26 (2019) 88 – 93.
- [11] M. Urgo, W. Terkaj, Formal modelling of release control policies as a plug-in for performance evaluation of manufacturing systems, *CIRP Annals* 69 (2020) 377 – 380.
- [12] K. Janowicz, A. Haller, S. J. Cox, D. L. Phuoc, M. Lefrançois, Sosa: A lightweight ontology for sensors, observations, samples, and actuators, *Journal of Web Semantics* 56 (2019) 1 – 10.
- [13] P. Dolog, Model-Driven Navigation Design for Semantic Web Applications with the UML-Guide, in: *Proc. of ICWE*, Munich, Germany, 2004, pp. 75–86.
- [14] T. M. de Farias, A. Roxin, C. Nicolle, Ifcwod, semantically adapting IFC model relations into OWL properties, *CoRR* abs/1511.03897 (2015). URL: <http://arxiv.org/abs/1511.03897>. arXiv:1511.03897.
- [15] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, N. Guarino, et al., Social roles and their descriptions., in: *KR*, 2004, pp. 267–277.
- [16] S. Borgo, E. M. Sanfilippo, W. Terkaj, Capabilities, capacities, and functionalities of resources in industrial engineering, in: *Proceedings of the Joint Ontology Workshops 2021 Episode VII: The Bolzano Summer of Knowledge co-located with the 12th International Conference on Formal Ontology in Information Systems (FOIS 2021), and the 12th International Conference on Biomedical Ontologies (ICBO 2021)*, Bolzano, Italy, September 11-18, 2021, volume 2969 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.
- [17] A. Galton, Operators vs. arguments: the ins and outs of reification, *Synthese* 150 (2006) 415–441.
- [18] P. M. Simons, *Parts: A study in ontology*, Oxford University Press, 1987.
- [19] F. Berardinucci, G. Colombo, M. Lorusso, M. Manzini, W. Terkaj, M. Urgo, A learning workflow based on an integrated digital toolkit to support education in manufacturing system engineering, *Journal of Manufacturing Systems* 63 (2022) 411–423.