International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: http://www.tandfonline.com/loi/tcim20

Feature-based product modelling: an ontological approach

Emilio M. Sanfilippo

INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING
https://doi.org/10.1080/0951192X.2018.1497814

ARTICLE

Feature-based product modelling: an ontological approach

Emilio M. Sanfilippo [a]

[a]Ecole Centrale de Nantes – Laboratoire des Sciences du Numrique (LS2N), UMR CNRS 6004, Nantes, France

ABSTRACT
Feature-based modelling is the leading approach for computer-based product design. The development of knowledge-based information systems brought to represent features in formalised ontologies. The lack of a clear ontological analysis of how features are understood in engineering led to formal models that treat features in an ambiguous manner, or reduce them to pure geometric elements. The purpose of the paper is to present a modular ontological architecture to support the explicit and machine treatment of features' semantic. Differently from the state of art, we provide a general framework where features are contextualised within a larger system for product knowledge representation. We focus on the fundamental (ontological) properties that features satisfy independently from specific applications and contexts of usage. The paper presents modelling case studies where the proposed ontologies are used to represent and (automatically) reason over product knowledge, and to analyse, compare, and integrate existing features classifications.

ARTICLE HISTORY
Received 25 August 2017
Accepted 11 June 2018

KEYWORDS
Feature-based modelling; features classification; ontology; semantic web

1. Introduction

Product knowledge representation and data management are knowledge intensive tasks carried out by means of computer systems, which allow experts to represent, share, and possibly integrate disparate quantitative and qualitative models. *Feature-based modelling* is the leading approach in these contexts, because of its capability to enrich Computer-Aided (CAx) models with information concerning modelling intents (Zhang et al. 2017; Ma, Chen, and Thimm 2008). A designer, for example, may want to declare a hole as being functional in the context of a product. This may be done by modelling the hole as a *functional feature*, that is, by characterising it by functional information. Differently, a manufacturer may be interested in the processes to be performed to realise the hole. In this sense, the hole is a *manufacturing feature* characterised, e.g. by machining information. Features can be thus used to represent multiple perspectives, and to support the integration of different stages of the product lifecycle (Chungoora, Canciglieri, and Young 2010).

Traditionally, features are represented via object-oriented methods for information management (see, e.g. Nasr and Kamrani 2007; Ma, Chen, and Thimm 2008). Nowadays the use of advanced computer systems calls for computational models with formal semantic to provide rigorous methods to share, integrate, and reason over heterogeneous data and knowledge in a way that is both transparent to experts and accessible to automated reasoners (El Kadiri and Kiritsis 2015; Chandrasegaran et al. 2013). This requirement brought to the application of knowledge engineering approaches and technologies, above all the exploitation of *ontologies* (Guarino, Oberle, and Staab 2009).

Despite relevant works, the development of feature-based ontologies has not been supported by a systematic approach or by an exploration of the various alternatives by which features can be interpreted and represented in computational terms. As a consequence, at the current state of art, it is unclear what features are with respect to a general conceptualisation of the engineering domain (Sanfilippo and Borgo 2016).

The purpose of the presented work is twofold. First, to analyse the notion of feature by means of ontology modelling principles (Guarino and Welty 2009). The goal is to understand what features are in a manner that is coherent with engineering knowledge and is contextualised within a larger knowledge representation framework. Second, on the basis of the analysis, to provide a representation of feature-based knowledge that is independent from specific application requirements and is therefore reusable. Differently from previous works we do not focus on algorithmic procedures or programming rules to recognise or specify features (see, e.g. Zhou et al. 2007; Zhang et al. 2017). We rather focus on the ontological properties by which features are characterised, and provide a logical representation to handle them in a machine-processable manner. Hence, the aim is not to present an ontology that axiomatises the features used in specific CAx systems. Rather, we set up an ontology that can be extended to meet specific application requirements.

The paper is structured as follows. In Section 2 we provide an overview of current approaches concerning feature knowledge representation by addressing their merits and limits. In Section 3 and Section 4 we present a modular ontology that is specifically targeted on the representation of features. The design principles behind the ontology are discussed together with an overview of some axioms. Section 5 provides a case study where the ontology is used to analyse, compare, refactor, and integrate existing features classifications. In the concluding section (Section 6) we highlight the differences and

CONTACT Emilio M. Sanfilippo emilio.sanfilippo@ls2n.fr Ecole Centrale de Nantes – Laboratoire des Sciences du Numrique (LS2N), UMR CNRS 6004, Nantes, France

benefits of our approach when compared with the literature. We also address the limits of the proposed ontology and open the way to future work. The Appendix shows how the ontology can be adopted to represent product and features knowledge, and to (logically) reason over it.

2. Feature-based information modelling: an overview

From the very beginning of feature-based approaches, features have been conceived in tight connection with cognitive phenomena and the need of abstracting from pure geometry in the practise of CAx-driven product development. In their seminal work, Shah and Mäntylä state that '[…] features are stereotypical knowledge structures embedded in cognitive processes in design, analysis, planning, and all other engineering activities […]' (Shah and Mäntylä 1995, p.13). Along the same line, Brunetti and Grimm (Brunetti and Grimm 2005) claim that '[t]he intention of introducing feature technology is to enrich CAx systems with knowledge structures similar to those used in human cognition to provide an additional layer of information making those systems more useful for design and to integrate design with downstream applications in the product life cycle, e.g. analysis, assembly, manufacturing, maintenance, recycling'.

These views are continuously recalled in the literature; e.g. a feature as 'the engineering significance of the geometry of a part' (Wingård 1991), 'a region of interest in a part model' (Bronsvoort and Jansen 1993), 'anything having a particular attribute of interest' (Usman et al. 2013). Expressions like 'engineering significance of the geometry of a part' or 'region of interest' mean that features represent technical knowledge in a manner that reflects experts' intents at a more abstract level than geometry. More explicitly, Rossignac (Rossignac 1990) introduces the notion of *intentional feature* as 'an abstraction of geometric elements'.

Despite their role, the adoption of feature-based approaches is hampered by the lack of transparent formal models to unambiguously represent, share, compare, and possibly integrate feature knowledge and data across communities and applications in a way that makes explicit and preserves experts' intents (Sanfilippo and Borgo 2016).

First, features are mostly used as CAx elements to enhance products' specifications. On the other hand, they are also understood as *physical* entities in space and time related to individual products. Shah and Mäntylä, for example, understand features as '*physical* constituent[s] of a part' (Shah and Mäntylä 1995, p.97). However, they also add that features are *information clusters* relevant to embed design intents into geometric models (Shah and Mäntylä 1995, p.10). In this way, they muddle the distinction between the physical and the information levels of feature modelling.

Second, although the emphasis of feature-based approaches is to abstract from geometric representations, features are commonly reduced to geometric specifications (see, e.g. Zhou et al. 2007; Zhang et al. 2016; Tang, Chen, and Ma 2013; Gupta and Gurumoorthy 2013). From this perspective one has to keep in mind that geometric formalisms do not allow for the explicit embedding of design intents into product models, e.g. functional or manufacturing intents (Guarino, Borgo, and Masolo 1997; Sanfilippo et al. 2017). This means that when features are reduced to geometric elements, they cannot be used to model the 'engineering significance' of products, to rephrase (Wingård 1991). For example, geometry is not suited to represent a slot feature as an entity that is intentionally designed with a functionality, or a protrusion as being made of the same material of the product to which it is related.

To overcome these limitations different communities have been investigating the exploitation of ontologies. As knowledge models with formal semantics, the core purposes of ontologies for product modelling are (at least) three (Chandrasegaran et al. 2013; El Kadiri and Kiritsis 2015). First, to represent experts' knowledge in a way that is transparent to third parties and is accessible to automated agents. Second, to disambiguate the meaning of the vocabulary used to organise and share data across information systems. Third, to allow machines to reason over knowledge and data in order to infer new facts over the stated axioms.

Unfortunately, the development of ontologies for feature-based product modelling is seldom guided by principled methodologies. As a result existing ontologies reckon on specific application requirements and are scarcely reusable without heavy re-engineering processes (see Section 5). Additionally ontologies for product modelling are commonly developed by translating engineering standards from their native specifications into logical languages (see, e.g. Barbau et al. 2012; Pauwels and Terkaj 2016) without investigating whether their conceptual and formal architectures are suited for ontological modelling.[1] The notion of machining feature in STEP-224 (ISO 10303 2006), for example, is ambiguous, since it refers to both a negative volume (void space) in a workpiece and the material removed for its creation. The encoding of STEP-224 in a language for ontology design does not help in solving the problem, whereas the two meanings of machining feature (negative volume vs. removed material) need to be clearly distinguished to foster the transparency of the data compliant with STEP.

In the next sections we introduce a modular ontology to overcome the issues identified through this section. In particular, Section 3 presents the Lightweight Upper Level Ontology (LUPO). This is extended in Section 4 to cover product-related notions resulting in the so-called Feature-based Product Ontology (FPRO). Both LUPO and FPRO are formalised in the Web Ontology Language (OWL) (W3C 2009), the W3C standard for Semantic Web ontologies. Recall that OWL is combined with Description Logics (Baader et al. 2003) to handle knowledge in a tractable manner, hence to allow for consistency checking, reasoning, and query answering procedures. The ontologies are developed by means of Protégé (version 5.2.0)[2] and checked against consistency with the HermiT reasoner (version 1.3.8). By means of the importing functionality of Protégé, FPRO is designed by importing LUPO and extending it with new classes, relations, and axioms. For reasons of space we do not show the entire axiomatisation of the ontologies; the reader can refer to the available OWL files for a detailed overview.[3]

3. Lightweight upper level ontology

LUPO is an upper-level ontology formalised in OWL hereby proposed as general semantic umbrella to support the development and harmonisation of domain and application ontologies. As we will see, the ontology is based on already existing resources, which are re-adapted for our purposes. Figure 1 shows the taxonomy of LUPO in the Unified Modelling Language (UML) notation.

The high-level distinction between materials, objects, processes, and qualities is based on (the core release of) the DOLCE ontology (Borgo and Masolo 2013), which has been already used for knowledge representation in design and manufacturing (see, e.g. Borgo and Leitao 2007; Solano, Romero, and Rosado 2016). These distinctions are also found in domain ontologies for product modelling (see, e.g. Štorga, Andreasen, and Marjanovic 2010; Usman et al. 2013; Fenves et al. 2008).

Qualities represent *characteristics*, e.g. the weight, shape, or space location of a driller, or the duration of a drilling operation. They are *dependent* entities in the sense that they must qualify other, non-quality entities in order to exist. For example, there cannot be a particular weight-quality q_1 on its own; in order to exist q_1 has to be the weight of, e.g. driller dm_1 . Following (Rijgersberg, van Assem, and Top 2013; Borgo and Masolo 2013), we assume that a quality cannot qualify multiple entities, e.g. q_1 can characterise only dm_1 . Consider now the drilling machine dm_2 (with $dm_2 \neq dm_1$); dm_2 bears its own weight-quality q_2 so that $q_1 \neq q_2$. Different qualities can however have the same *value*; e.g. both dm_1 and dm_2 are 5 kg heavy, namely, their qualities, q_1 and q_2, have the same value which is measured with the same measurement scale. The distinction between qualities and their values is useful to model the fact that qualities can undergo value changes, as well as the fact that a value can be provided according to diverse (and possibly comparable) measurement systems.

The specification of qualities in LUPO, in particular the distinction between qualities, values, and measurement scales is based on the *Ontology of Units of Measure and Related Concepts* (Rijgersberg, van Assem, and Top 2013). By (Ax1), qualities characterise (QUALITYOF)[4] only objects, processes, or materials. The cardinality restriction attached to QUALITYOF is exactly1 meaning that each instance of QUALITY characterises only one individual entity.

```
Ax1 Class: QUALITY
  SubClassOf
    QUALITYOF only (OBJECT or PROCESS or MATERIAL)
    QUALITYOF exactly1 (OBJECT or PROCESS or MATERIAL),
```

The (data property) relation hasNumericalValue and the (object) relation hasUnit, which are both imported with their native URIs from Rijgersberg, van Assem, and Top (2013) can be used to specify the value and measurement unit of qualities, respectively. The range of the former relation can be thus an integer or a float, among others, whereas the range of the latter is an instance of the UNIT class, e.g. *Kilogram* for weight qualities.[5] For instance, (f1) – (f2)[6] represent the weight-qualities q_1 and q_2 characterising (the drillers) dm_1 and dm_2, respectively, such that q_1 and q_2 have numerical value 5 and measurement unit *Kilogram*, i.e. they have a value of 5 kg.

```
f1 Individual : q1                f2 Individual : q2
  Types                             Types
    WEIGHTQUALITY                     WEIGHTQUALITY
  Facts                             Facts
    QUALITYOF dm1                     QUALITYOF dm2
                                        HASNUMERICALVALUE 5
  HASNUMERICALVALUE 5                   HASUNIT Kilogrom
    HASUNIT Kilogrom                Different From
                                        q1
```

In formulas (f1) – (f2), WEIGHTQUALITY stands for a subclass of QUALITY like SPATIALQUALITY and TEMPORALQUALITY in Figure 1, representing spatial and temporal locations, respectively. Specific quality kinds can be introduced according to application requirements, e.g. by importing classes from Rijgersberg, van Assem, and Top (2013) under the basic taxonomy of LUPO.

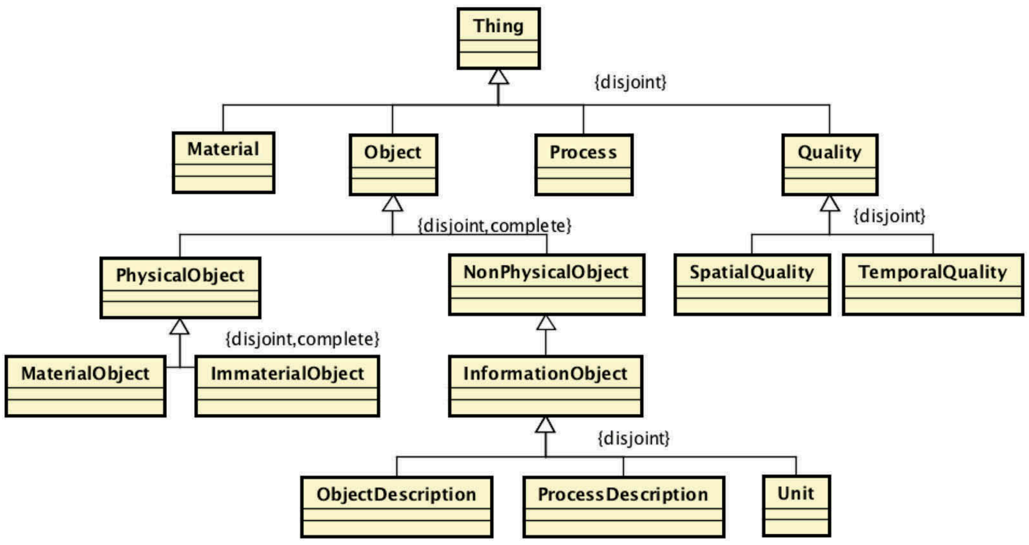

Figure 1. Taxonomy of LUPO.

Processes are entities that are primarily characterised by temporal qualities and have objects as participants, see (Ax2). The relationship of *participation* (hasParticipant) is the most general link between processes and objects (or materials); it can be however extended to cover more specific relations in the manufacturing domain, e.g. input–output relations in the spirit of IDEF0.[7]

Ax2 Class: Process
  SubClassOf:
    hasParticipant only (Object or Material),
    hasParticipant some Object,
    hasParticipant some TemporalQuality

Moving to objects, by (Def1) we distinguish between the (disjoint) classes PhysicalObject and NonPhysicalObject (see Figure 1), where only instances of the former are located in space. PhysicalObject specialises in the partition covering MaterialObject and immaterialObject. As the terminology suggests, differently from the latter, instances of the former are made of some materials, see (Def2) and (Def3). For example, gear *gr* is made of metal *ml* , differently from the cavity *cv* in *gr* , which is not made of any material; *cv* is therefore an immaterial object. As we will see in the next section, the representation of immaterial objects is useful to model the difference between additive and subtractive features.

Def1 Class: Object
  Equivalent To:
    (PhysicalObject or NonPhysicalObject)
Def2 Class: MaterialObject
  Equivalent To:
    PhysicalObject
    and (madeOf Some Material)
Def3 Class: ImmaterialObject
  Equivalent To:
    PhysicalObject,
    and (not (madeOf some Material)

The class NonPhysicalObject is the most general classifier for objects lacking spatial locations. Among them, information objects (also called descriptions) refer to the 'content' of documents. An example is the content of a CAD model representing the geometric properties of a certain product. Following the CIDOC ontology (Crofts et al. 2008), we assume that information objects can be carried in different supports. For instance, the same CAD information object can be carried in two or more STEP files, or paper sheets. To grasp the fact that an information object can be about physical entities like products or processes, we distinguish between ObjectDescription and ProcessDescription as showed in Figure 1. An example of the former is a CAD information object; an example of the latter is a process plan. When a specific entity, e.g. a product, complies with a description, i.e. it complies with the properties specified in the description; we say that it *satisfies* the description.

Table 1 reports the relations used in lupo. Apart from hasValue, which – as we saw – is an OWL data property, all the other ones are OWL object properties. For each relation the table shows the inverse, when present. When multiple classes appear in either the domain or range columns, a disjunctive reading is assumed. Note that the relation of *proper part* is relevant to model parthood relations, e.g. a drilling bit that is part of a driller, or an activity of drilling that is part of a larger manufacturing process. From a formal perspective, proper parthood is a strict order, i.e. it is irreflexive, asymmetric, and transitive (Casati and Varzi 1999). However, given the expressivity of OWL, *proper part* is only transitive in lupo. Following DOLCE (Borgo and Masolo 2013), we assume that parthood is inter-categorial, namely, it holds exclusively between the instances of a given class, namely, between only objects, or processes, or materials, see (Ax3) – (Ax5). In the case of objects, *proper part* is further restricted between either non-physical or physical objects. Also, axioms (Ax6) – (Ax7) exclude material objects to have immaterial objects as parts, and vice versa. In the next section, we will see how *proper part* is extended to the engineering domain.

Ax3 Class: Object
  SubClassOf:
    hasProperPart only Object
Ax4 Class: Material
  SubClassOf:
    hasProperPart only Material
Ax5 Class: Process
  SubClassOf:
    hasProperPart only Process
Ax6 Class: MaterialObject
  SubClassOf:
    hasProperPart only MaterialObject
Ax7 Class: ImmaterialObject
  SubClassOf:
    hasProperPart only ImmaterialObject

4. Feature-based product ontology

In this section we present the extension of LUPO to the engineering domain resulting in the FPRO. Since our aim is to lay down the ontological foundations of feature-based modelling approaches, we will focus on the representation of features within a larger system for product knowledge representation.

We first start with the notion of product, which is notoriously an ambiguous term due to its various meanings (Borgo

Table 1. Relationships in LUPO.

Relation	Domain	Range	Example
HasQuality (QualityOf)	Object, Material, Process	Quality	Driller *dm* has weight-quality *q*
HasValue	Quality	Datatype	Quality *q* has value 5 (integer)
HasUnit	Quality	Unit	Quality *q* has unit *Kilogram*
HasParticipant (ParticipatesIn)	Process	Object, Material	Process *p* has participant John (object)
HasProperPart (ProperPartOf)	Object, Material, Process	Object, Material, Process	Driller *dm* has proper part drilling bit *b*
MadeOf (Makes)	Object	Material	Gear *gr* is made of metal *mt*
Satisfies (SatisfiedBy)	Object, Material, Process, Quality	Information Object	Gear *gr* satisfies object description *ds*

et al. 2014). For our purposes we restrict to the domain of engineering design and rely on the conceptualisation of products as *designed* human-made objects. This is a common view across information modelling resources to refer, for instance, to machining tools or the items they manufacture (see, e.g. Usman et al. 2013; Štorga, Andreasen, and Marjanovic 2010; Solano, Romero, and Rosado 2016). In order to facilitate the reuse of FPRO across engineering domains, Product is introduced as a primitive class subsumed by PhysicalObject that each user can (consistently) characterise according to specific requirements.

Product is specialised in MaterialProduct to cover designed objects that are made of matter.[8] This latter class covers *assembled* and non-assembled, hereby called *single*, products, see Figure 2.[9] As the terminology suggests, differently from assembled products, single products are not formed by any other product, namely, they do not have any component, see (Def4) and (Def5).[10] The hasComponent relation used in the definitions extends hasProperPart (see Table 1) and holds only between products (or physical features, as we will see). In this manner we treat the latter as the most general structural relation, whereas the former can be used in a more specific manner when the relata are both products (or physical features). This approach allows to distinguish the notions of *having parts* and *having components*. For example, one may want to talk about the parts of a screw like the head or body, while treating the screw as a product with no components (hence, as a single material product). In this sense the head and body are parts but not components of the screw.

Def4 Class : SingleMaterialProduct
  Equivalent To:
    Material Product,
    and (not (hasComponent some Product))
Def5 Class: AssembledMaterialProduct
  EquivalentTo:
    MaterialProduct,
    and (hasComponent some Product),
    and (hasComponent min 2 SingleMaterialProduct)

As said in the previous section, objects may satisfy descriptions. For engineering modelling purposes, it is relevant to model explicitly the fact that a product complies with a corresponding design. In our ontology this is grasped by the class TechnicalMaterialProduct.[11] By (Def6) its instances satisfy product descriptions (information objects) and are created by creation processes. A detailed representation of manufacturing processes is behind the purposes of this work. A creation process is (weakly) understood as a process that has objects, materials, or qualities as outcomes, see (Def7). As it can be seen for the definition, creation processes do not necessarily comply with (process) descriptions, hence their outcomes may not correspond to the desired ones.

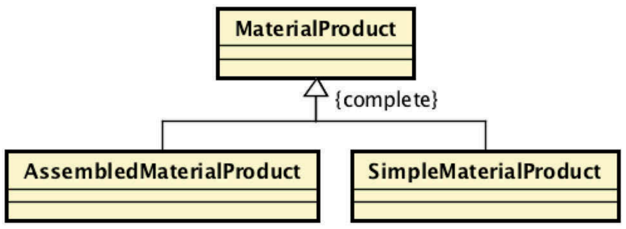

Figure 2. Taxonomy of material product.

Def6 Class : TechnicalMaterialProduct
  EquivalentTo:
    MaterialProduct,
    and (satisfies some ProductDescription),
    and (outcomeOf some CreationProcess),
    and (outcomeOf only CreationProcess)
Def7 Class : CreationProcess
  EquivalentTo:
    Process,
    and (hasOutcome only (Material only Object or Quality)),
    and (hasOutcome some (Material only Object or Quality))

We can now move to the representation of features. Recall from Section 2 that we identified two core interpretations for the notion of feature, namely, (i) information clusters for embedding experts' intents into geometric models, and (ii) (specific types of) physical entities related to products. Following Sanfilippo and Borgo (2016), we call *information features* (I-features) the former, and *physical features* (P-features) the latter. The two notions are strictly related since P-features are meant to satisfy (comply with) the properties established at the I-feature level. The notions cannot be however confused: I-features, as Brunetti (2003) would claim, exist only within product descriptions, whereas P-features are entities of the physical world. Clearly, given an I-feature, its corresponding P-feature(s) may not exist, since the former may not be physically realised. This is common in design and manufacturing contexts where the creation of product models does not always lead to production.

Following this reasoning, Feature is the most general classifier we use; it is directly subsumed by Object and covers both PFeature and IFeature. The latter class is (weakly) characterised as a subclass of ObjectDescription, hence its instances are information objects; further constraints can be added to grasp specific knowledge. The former is characterised by (Ax8), where the relationship featureOf expresses the most general link associating a feature to a non-feature physical object. Also, this relation establishes a sort of existential dependency between P-features and the objects to which they relate. Accordingly, a P-feature, e.g. a hole or bump, cannot exist if not related to something which in turn is not a P-feature.[12] Note that a similar dependency constraint does not hold for I-features which can be specified without being related to any other information object.

Ax8 Class : PFeature
  SubClassOf:
    FeatureOf only PhysicalObject,
    FeatureOf some PhysicalObject,
    not (FeatureOf some PhysicalFeature)

PFeature specialises in MaterialPFeature, VoidPFeature, ElementaryPFeature, and CompoundPFeature, see the taxonomy in Figure 3. As we will see, the distinction between the first two classes is based on material properties, whereas the latter two classes are distinguished because of their structure.

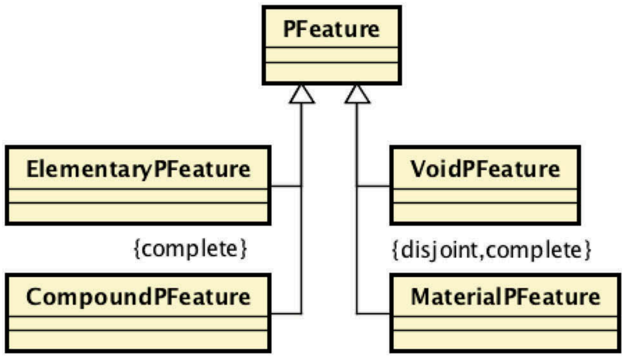

Figure 3. Taxonomy of PFeature.

As the labels suggest, MaterialPFeature (e.g. protrusion, bumps) is used for features that are made of material, differently from instances of VoidPFeature (holes, steps, pockets, etc.) which lack material constitution, see (Def8) and (Def9).[13] This dichotomy reflects the distinction between *additive* and *subtractive* features, respectively, which is common in the literature (Shah and Mäntylä 1995).[14] Because of the formal definitions, OWL reasoners are able to classify MaterialPFeature and VoidPFeature as MaterialObject and ImmaterialObject, respectively.

Def8 Class : MaterialPFeature
EquivalentTo:
PhysicalFeature
and (madeOf some Material)

Def9 Class : VoidPFeature
EquivalentTo:
PhysicalFeature
and PhysicalObject
and (not (madeOf some Material))

The following rule, written in the W3C Semantic Web Rule Language (SWRL) (Horrocks et al. 2004), is used to increase the expressivity of the ontology and rule out undesired interpretations.[15]

R1 featureOf(?x, ?y), MaterialPFeature(?x) → properPartOf(?x, ?y)
(if material P-feature x is feature of y, then x is proper part of y)

First, according to (R1), a material P-feature is proper part of the object to which it is related via featureOf. Second, we know from the previous section that the relation of proper parthood is restricted to either material or immaterial objects, see (Ax7) and (Ax6). Looking at (R1), this means that the variable *y* in FeatureOf $(?x, ?y)$ refers to a material object. Together with the disjointness axiom between material and immaterial objects (see Figure 1), it follows that material features cannot be features of immaterial objects. This is reasonable from both an ontological and an engineering viewpoint, since a material entity like a protrusion cannot be the feature of something immaterial. Assessing formulas like (f3) and (f4) would thus generate an inconsistency in our ontology. Third, given the distinction between the relations properPartOf and componentOf we saw above, a material feature is proper part of the product to which it relates, although it is not one of its components. This is reasonable from an engineering modelling stance, since features do not qualify as products' components.

f3 Individual: f_1
Types:
MaterialPFeature
Facts:
featureOf pob_1

f4 Individual : pob_1
Types:
ImmaterialObject

Along the line of (R1), rule (R2) constrains the relations featureOf and properPartOf in the case of void features. Accordingly, only when a void feature is feature of an immaterial object, it qualifies as proper part of the object.

R2 featureOf (?x, ?y), VoidPFeature (?x), ImmaterialObject (?y) → properPartOf (?x, ?y)
(if void P-feature *x* is feature of immaterial object *y* , then *x* is proper part of *y*)

With this machinery at hand we now model the distinction between *elementary* and *compound* features as found in engineering (Shah and Mäntylä 1995).[16] Following (Bronsvoort and Jansen 1993) elementary features cannot be further decomposed into simpler features. Differently, compound features are composed of several features. An example of the latter is a stepped void-hole resulting from the aggregation of two concentric (elementary) void-holes. In our approach, ElementaryPFeature is defined as a P-feature that has no other P-features as components, see (Def10). Conversely, a compound feature is composed by some P-features amongst which at least two elementary P-features (Def11).

Def10 Class : ElementaryPFeature
EquivalentTo:
PFeature
and (not (hasComponent some PFeature))

Def11 Class : CompoundPFeature
EquivalentTo:
PFeature
and (hasComponent some PFeature)
and (hasComponent min 2 ElementaryPFeature)

Note from Figure 3 that the material and structural characterisations of P-features are not disjoint. Hence, both characterisations can be (consistently) combined. For example, one may want to represent a compound void or material P-feature (see Appendix). The ontology has already the expressivity to represent P-feature compositions with respect to material and structural properties. Recall that the relation hasComponent is subsumed by hasProperPart, hence componenthood is restricted between either material or immaterial objects, too. Accordingly, material and void features compose exclusively with material and void features, respectively. Following this reasoning, (f5) – (f6) generate an inconsistency in our ontology, hence they are ruled out.

f5 Individual : f_1
    Types:
        MaterialPFeature
    Facts:
        ComponentOf f_2

f6 Individual : f_2
    Types:
        VoidFeature

Generalising from both material and structural properties, (Def12) replaces (Ax8) and defines PFeature as being equivalent to either material or void P-features, elementary or compound P-features.[17] Axiom (hasComponent only PFeature) guarantees that P-features compose only between them.

Def12 Class : PFeature
    EquivalentTo:
        MaterialPFeature or VoidPFeature,
        ElementaryPFeature or CompoundPFeature
    SubClassOf:
        Feature,
        featureOf only PhysicalObject,
        featureOf some PhysicalObject,
        hasComponent only PFeature,
        not (featureOf some PFeature)

Finally, (R3) and (R4) establish some constraints between the notion of product, feature, and component. Table 2 reports the relations introduced in FPRO.

R3 Product(?x), hasComponent(?x,?y), hasFeature(?y,?z) → hasFeature(?x,?z)
    (if product x has component y with P-feature z , then z is feature of x , too)

R4 Product(?x), hasComponent(?x,?y),hasFeature(?y,?z), componentOf(?z,?v) → hasFeature(?x,?v)
    (if product x has component y with P-feature z such that z is (feature) component of v , then v is feature of x)

In the next section we will see how both lupo and fpro can be used to analyse, compare, and integrate different features classifications. This will also show how our ontology can be extended with feature classes that are recurrent in the literature. In the Appendix section we show the use of the ontologies to represent products and features, and to reason over them.

5. Analysing and integrating features classifications

Features can be classified according to disparate principles, e.g. based on geometric and topological constraints (Poldermann and Horváth 1996; Han 1996; Fontana, Giannini, and Meirana 1999), manufacturing methods of production (Han, Pratt, and Regli 2000; Case and Wan Harun 2000), or functional properties (Schulte, Weber, and Stark 1993; van Holland and Bronsvoort 2000), just to mention a few approaches. Also, experts agree that classifications are hardly exhaustive since feature classes can be extended and customised to meet contextual requirements (Han, Pratt, and Regli 2000).

Multiple classification criteria are useful to embed experts' intents into feature models. On the other hand, however, by representing features only with respect to specific perspectives, hence without a broader view on how multiple perspectives integrate, classifications become highly fragmented. The risk is that, first, each classification remains isolated within its own contextual assumptions and application goals. Second, consistency among multiple classifications is not guaranteed, since their assumptions may clash against each other. Third, data exchange and interoperability between information systems and communities is impeded, because the meaning attached to feature data can significantly vary. To exemplify the discussion, we consider the taxonomies in Figures 4 and 5, which are both taken from the literature (Gupta and Gurumoorthy 2013; Zhang et al. 2017).[18]

Features are classified in Figure 4 on the basis of geometric and topological principles. According to its authors (Gupta and Gurumoorthy 2013), the taxonomy provides a unified perspective that puts together volumetric, deformation, and free-form features. The first ones are associated with addition or subtraction of volume; the second ones are created by deforming (a part in) a product, and the third ones are attached to freeform (products') surfaces typically modelled with NURBS or similar methods (see, e.g. Fontana, Giannini, and Meirana 1999; van den Berg, Bronsvoort, and Vergeest 2002). As it can be seen from the figure, apart from the classification of features like Hole or Protrusion, both VolumetricFeature and FreeFromSurfaceFeature cover four types of shape attributes, namely, Blind, Closed, Through, and DoubleBlind.

Differently from this approach, the classification in Figure 5 is based on manufacturing principles (Zhang et al. 2017). At the most general level, MachiningFeature refers to features that are created by machining processes. The class is extended into 2.5DFeature and 1DFeature to distinguish between features obtained with different machining techniques. These two classes are further specialised to cover common examples of machining features, e.g. ThroughSlot or BlindHole, which are then characterised via geometric and topological constraints.

It should be clear that the taxonomies provide two alternative views on features. As a consequence, they have some classes in common although these do not share the same intended semantic. Additionally, they both suffer from ontological deficiencies when looked into details.

First, in both cases, it is not clear from the models what features are with respect to a larger view on the engineering domain. The classifications rely on users' background

Table 2. Relationships in fpro.

Relation	Domain	Range	Example
hasFeature (featureOf)	PFeature	MaterialObject, ImmaterialObject	Gear *gr* has feature hole *hl*
hasComponent (componentOf)	PFeature, Product	PFeature, Product	Counterbore hole *ch* has (feature) component hole hl_1 and hl_2
hasOutcome (outcomeOf)	Process	Material, Object, Quality	Process *p* has outcome product *pr* with feature hole *hl*

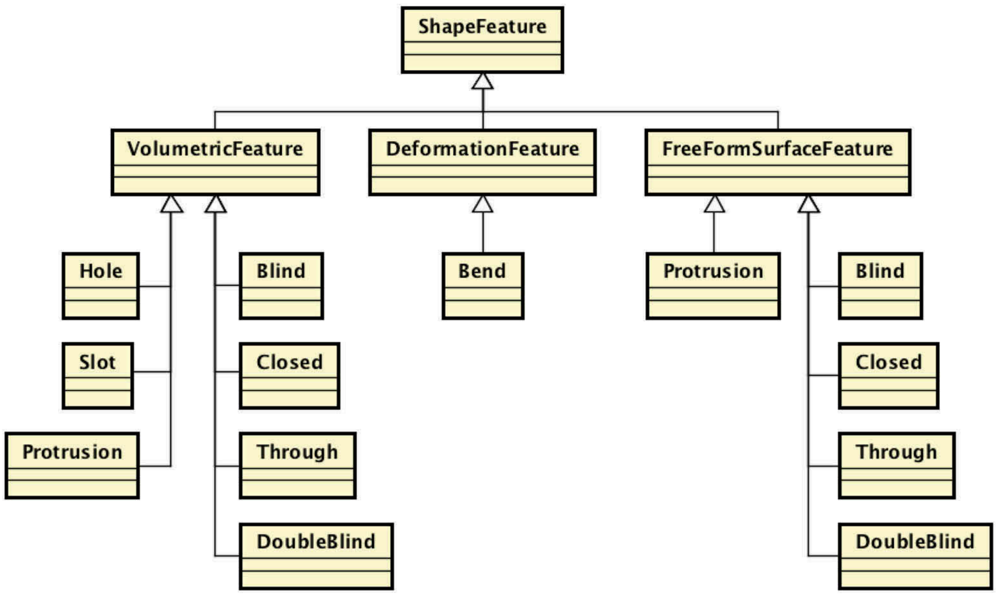

Figure 4. Shape feature taxonomy based on Gupta and Gurumoorthy (2013).

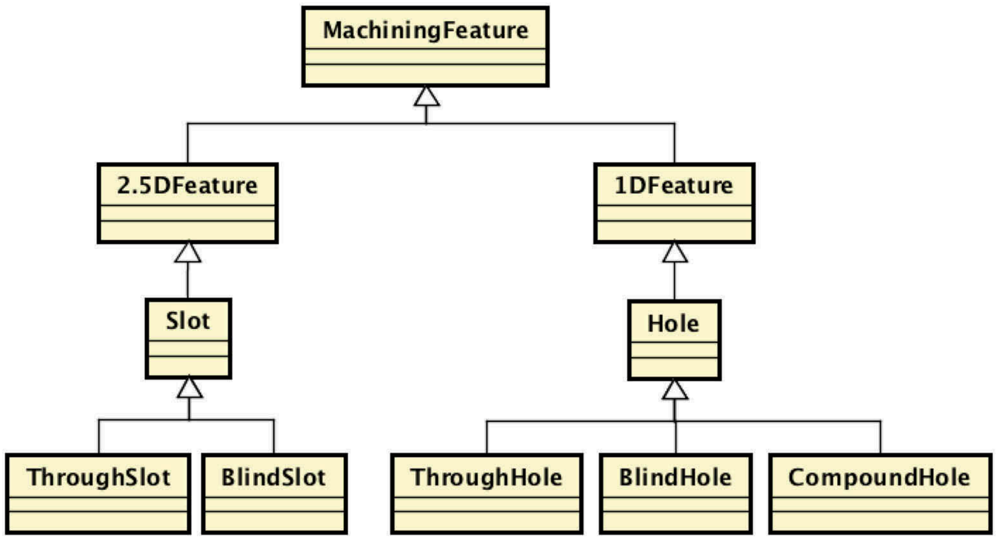

Figure 5. Machining feature taxonomy based on Zhang et al. (2017).

knowledge for the interpretation of their classes. Hence, they are both underspecified with respect to both engineering knowledge and ontological modelling principles. For example, it is not clear whether instances of ShapeFeature (see Figure 4) and MachiningFeature (see Figure 5) are CAx modelling elements (I-features in our terminology), or features in physical products (P-features). This is at the expenses of both human understanding and machine reasoning.

Second, by looking at Figure 4, the taxonomy mixes shape and operational information. Compare FreeFromSurfaceFeature and DeformationFeature. Instances of the former class are purely geometrically characterised, whereas instances of the latter are characterised with respect to deformation processes that modify the geometry of the product at stake. Also, the taxonomy covers classes like Blind or Through as types of features, whereas – from an ontological perspective – they refer in first instance to features' geometric properties. Note that these classes are duplicated in the taxonomy whereas their representation as geometric properties allows to avoid the duplication.

Third, looking at Figure 5, we saw that the proposed taxonomy is meant to classify features according to manufacturing knowledge, whereas – as a matter of fact – its classes are only geometrically and topologically characterised. MachiningFeature and its subclasses are informally understood as features resulting from machining operations. Their formal specification should hence cover machining knowledge while this is not actually the case.

Figure 6 shows the (partial) refactoring and integration of the taxonomies in Figures 4 and 5 under (a portion of) our ontology. For the sake of the example we limit ourselves to the representation of physical features, whereas their corresponding information features can be easily introduced. Looking at Figure 6, at the general level, we reuse and extend the classes PFeature and

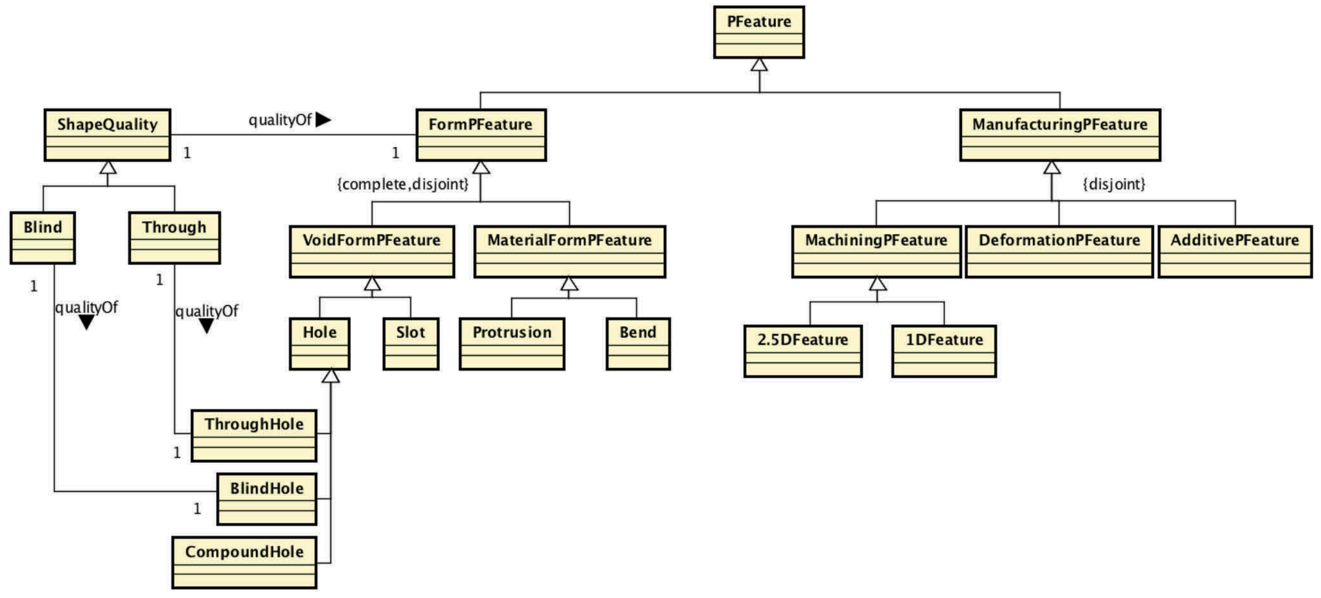

Figure 6. Integration of feature taxonomies.

ShapeQuality. The latter is specialised into Blind, Through, Closed, and DoubleBlind (the latter two classes are not shown in the UML taxonomy) in order to model shape attributes and characterise instances of PFeature. This class is now specialised in FormPFeature and ManufacturingPFeature.

A form P-feature is defined as a P-feature with a shape quality, independently from its structure or material properties. From this perspective, FormPFeature stands for ShapeFeature in Figure 4.[19] The class is then specialised to cover VoidFormPFeature and MaterialFormPFeature, which can be easily defined in the FPRO ontology, see (Def13) and (Def14), respectively.

Def13 Class : VoidFormPFeature
EquivalentTo:
FormPFeature and VoidFormPFeature
Def14 Class: MaterialFormPFeature
EquivalentTo:
FormPFeature and MaterialPFeature

Classes like Hole, Slot, Protrusion, and Bend can be now characterised with respect to material properties, whereas this is done neither in (Zhang et al. 2017) nor (Gupta and Gurumoorthy 2013). Accordingly, Hole and Slot, Protrusion and Bend are classified under disjoint branches of the taxonomy (see Figure 6). The cut-off distinction between shape qualities and form P-features allows to represent classes like BlindHole or ThroughSlot (not shown in Figure 6) in a clear manner and without duplicating shape qualities. Also, our ontology provides a formal manner to define CompoundHole, see (Def15), which remains a primitive notion in (Zhang et al. 2017).

Def15 Class : CompoundHole
EquivalentTo:
Hole and CompoundPFeature

To make sense of manufacturing knowledge, the class ManufacturingFeature is defined as a P-feature that results from a manufacturing process (a subclass of Creation Process in our approach). The class is then extended into MachiningPFeature, DeformationPFeature, and AdditivePFeature. These are defined as physical features resulting from machining, deformation, and (material) additive processes, respectively; see (Def16) for an example of definition. Note that FormPFeature and ManufacturingPFeature are not disjoint (nor they form a complete specialisation of PFeature), hence we may have an instance of, e.g. Bend that is both a MaterialFormPFeature and a DeformationPFeature. In this manner we are able to cover the intended semantics of Figure 4, where Bend is classified as a deformation feature. However, differently from Gupta and Gurumoorthy (2013), we now explicitly state the distinction between features that are geometrically characterised and features that satisfy operational constraints.

Def16 Class : DeformationPFeature
EquivalentTo:
PFeature
and (outputOf some DeformationPFeature)
and (outputOf only DeformationPFeature)

Finally, note that the classes VolumetricFeature and FreeFormFeature present in Figure 4 are not included in our taxonomy. If the semantics of the former is spelled out in pure geometric terms as, e.g. in Sakurai and Chin (1994), it can be introduced in our taxonomy as a subclass of FormPFeature properly characterised with respect to shape qualities. The latter class can be easily defined as a (either material or void) form feature that is characterised by a free-form shape quality.

To summarise, we showed in this section how our (modular) ontology can be used to analyse, compare, restructure, and (possibly) integrate existing features classifications. The purpose is to increase the transparency and stability of classifications, as well as to provide a unified framework of the concepts at stake. Looking at Figure 6, FormPFeature and ManufacturingPFeature can be now used to provide alternative yet well-integrated perspectives on features. The taxonomy can be enhanced with further feature classes at different level of generality, e.g. FunctionalPFeature can be added to model explicitly functional information. Also, shape qualities can be characterised via topological and geometric constraints. However, given the limited expressivity of Semantic Web languages and the formal complexity of logic-based geometric modelling (Borgo and Masolo 2010), we prefer avoiding a logical treatment of shapes. These can be handled by the standard methods of computational geometry (Stroud and Nagy 2011) used in tandem with ontologies for a qualitative treatment of shapes and an explicit representation of non-geometric information.

6. Discussion and conclusion

We discussed throughout the paper the ontological representation of product knowledge with an explicit focus on feature-based product modelling. The overall purpose is to provide an ontology for the transparent and machine-processable management of feature-based data.

In order to foster the applicability and reuse of our work, we developed a modular architecture comprising two ontologies. The first one, LUPO is a (light-weight) upper-level ontology whose design principles are based on existing reference ontologies, as well as data modelling standards. It is independent from specific application requirements, therefore it can be extended and exploited for various tasks. The second one, FPRO, is based on LUPO and is specifically targeted on the representation of products and features. The latter two notions are both specialised by taking into account material and structural properties. Concerning products, we distinguished between single and assembled products, material and immaterial products. Similarly, features can be either material or void, elementary or compound. Additionally, in both LUPO and FPRO we established a cut-off distinction between information objects and their physical counterparts, when they exist. This approach allows to distinguish in a systematic manner between, e.g. CAx models and the corresponding physical products (or features). Accordingly, features can be either I-features or P-features. Also, we formalised (in the expressivity of OWL) the distinction between the relations of proper-part and component-of, which are useful to distinguish the parts of a product from its components.

To the best of our knowledge the characterisation of features in terms of their material, composition, and dependency conditions, as well as the formal treatment of the distinction between I-features and P-features are novelties with respect to the state of art about ontology-based feature modelling. Differently from feature-based models like Tessier and Wang (2013), Tang, Chen, and Ma (2013), Romero, Rosado, and Bruscas (2015), Gupta and Gurumoorthy (2013), Usman et al. (2013), Zhang et al. (2017), among others, our work has to be understood as a *foundational* work aimed at making explicit and formalising *fundamental* properties that characterise features independently from their context of usage. For instance, independently from the fact that a bump is perceived as a functional or manufacturing feature, if it exists in the physical world, then it is made of some material (possibly the same of the product to which it relates), may result from the composition of other (material) features, and cannot exist if not ultimately related to a non-feature (material) object. All these properties are commonly assumed in the engineering literature, although they are not made explicit in current formal models and ontologies.

It should be clear that FPRO is not meant to provide a *complete* representation of the classes and relations needed for feature-based modelling in specific scenarios. As said, it rather provides general elements to be further extended. An example is showed in Section 5 where different types of both form and manufacturing features are introduced. Additionally, in the same section we showed how LUPO and FPRO can be used to analyse existing features classifications and integrate them. An integrated treatment of multiple modelling perspectives can facilitate the comparison of different information systems and enable data sharing among them. Note that in both Tessier and Wang (2013) and Tang, Chen, and Ma (2013), the authors introduce general models to represent features across applications in a similar manner. What they propose are *data models* where a generic feature class is specified as the aggregation of multiple attributes, mathematical parameters, geometric, and topological constraints. They do not clarify what features are with respect to an overall understanding of the engineering domain. Also, the proposed models cannot be used to analyse and compare the classifications presented in Section 5 (see Figures 4 and 5), since they lack the conceptual tools to make sense of basic ontological distinctions, e.g. I-features vs. P-features, material features vs. void features, or form features vs. manufacturing features. To be more precise, in the case of Tang, Chen, and Ma (2013), the authors showed how their approach can be extended and applied to various engineering domains (see, e.g. Uddin and Ma 2015; Yusuf and Ma 2016). Despite this, the ontological properties of feature classes are not made explicit. By looking at Uddin and Ma (2015) and Yusuf and Ma (2016), it remains, e.g. unclear whether (physical) features can exist by themselves, or whether the features of a product *p* propagate to the larger product of which *p* is component of. As we saw throughout the paper, making explicit these assumptions is fundamental to disambiguate the intended semantic of feature notions and to allow machines to reason over knowledge and data.

From an application perspective FPRO – once extended with specific task-driven elements – can be employed for various purposes; some examples follow. First, it can be integrated within a Product Lifecycle Management (PLM) software suit to organise heterogeneous feature data coming from different applications, e.g. Computer-Aided Design (CAD) and Computer-Aided Process Planing (CAPP) systems, among others. In this scenario the ontology acts as common framework to integrate multiple data in the same repository. This is particularly relevant in emerging paradigms and approaches like Industry 4.0 (Smart Manufacturing) (Weichhart et al. 2016) and Cloud Manufacturing (Lu, Morris, and Frechette 2016) where various applications are required to share data and knowledge in a seamless manner (see also Mourtzis and Doukas 2012; Efthymiou et al. 2015). Second, if the data models of each application in the PLM suit are aligned to the ontology, there is a higher chance for the applications to (semantically) interoperate, since the ontology provides the common language among them. Third, the embedding of

FPRO in an organisation's information system can support decision-making procedures. For example, a knowledge base developed on the grounds of the ontology can be queried to retrieve useful information and, hence, to take advantage from the organisation's experience stored in the knowledge base (for a similar approach, see, e.g. Efthymiou et al. 2015). Fourth, the reasoning capabilities of FPRO (see Appendix) can be helpful to detect bottlenecks in engineering data. For instance, specific rules can be set up to define the dimensions that certain features have to satisfy if they are meant to carry assembly functionalities. A reasoner can thus warn experts when dimensional constraints are violated in the models.

In both LUPO and FPRO spatial information, e.g. geometry, cannot be expressed with the same expressivity of quantitative approaches. For instance, we cannot represent *concentric* void-holes or *parallel* surfaces. As said by the end of Section 5, this is due to the use of computational logic for knowledge representation and its emphasis on qualitative, rather than precise, mathematical modelling. A way of dealing with computational spatial modelling in combination with tractable ontologies may lead to the development of a 'hybrid' modelling system combining symbolic methods with geometric modelling techniques commonly used in engineering (e.g. B-rep, mesh modelling, etc.). In this view the ontology would allow for the qualitative representation of the concepts at stake, while a set of geometric elements would be used to enrich the instantiation of the ontology with the required quantitative (geometrical, topological, etc.) constraints. The development of such a hybrid system requires however future work. Additionally, further work is necessary to enhance the applicability of the ontologies to the manufacturing domain, where features play a relevant role to integrate multiple knowledge. From this perspective, FPRO needs to be combined with an ontology for manufacturing processes and resources. This would allow to model features in tight connection with the operations and machines employed for their production.

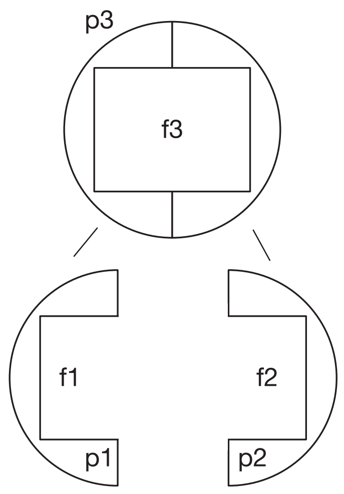

Figure 7. Examples of physical products with P-features.

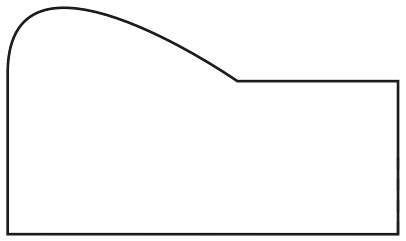

Figure 8. Example of physical block with material bump feature.

Notes

1. The reader can refer to Sanfilippo and Borgo (2016) for an overview of the limitations of current ontologies and conceptual models for feature-based product modelling.
2. http://protege.stanford.edu/, last access March 2018.
3. Both ontologies are available at: http://www.loa.istc.cnr.it/ontologies/LupoLib.zip.
4. The relationship QUALITYOF corresponds to *inherence* in DOLCE (Borgo and Masolo 2013). Also, the overall approach hereby adopted to represent qualities is a simplified version of DOLCE's approach tuned to the expressivity of OWL.
5. Units are imported from Rijgersberg, van Assem, and Top (2013), too.
6. We prefix formulas with *f* for examples.
7. See http://www.idef.com/, last access March 2018.
8. Along the same lines a class for *immaterial* products like softwares is introduced in the ontology.
9. Single products are often called *parts* in the literature (Sanfilippo et al. 2016). From Figure 2 note that ASSEMBLEDMATERIALPRODUCT and SINGLEMATERIALPRODUCT exhaust the domain of material products hereby considered. These classes are not disjoint since an assembled product may be defined as the (ordered) aggregation of single products.
10. Axiom (HASCOMPONENT min 2 SINGLEMATERIALPRODUCT) in (Def5) guarantees that the decomposition of assembled products always terminates in single products.
11. The label *technical* product is borrowed from Borgo et al. (2014).
12. In the Industry Foundational Classes (IFC) standard features are understood as 'existence dependent elements', see IFCFEATUREELEMENT on http://www.buildingsmart-tech.org/, last access on March 2018.
13. The label *void* feature is taken from IFC (see IFCVOIDINGFEATURE). Void features are also called *negative* or *clear volumes* across the literature (Sanfilippo and Borgo 2016).
14. The reader can refer to the specialisation of IFCVOIDINGFEATURE in IFC.
15. See Papakostas et al. (2014), Efthymiou et al. (2015), and Rentzos et al. (2012), among others, for modelling approaches using rules in connection with ontologies for the engineering domain.
16. The terminology may vary; e.g. elementary features are also called *simple* or *atomic* features (Bronsvoort and Jansen 1993; Nasr and Kamrani 2007).
17. Note that, from a logical perspective, the SubClassOf axioms in (Def12) should be better used to characterise the subclasses of PFEATURE. We decide to include them at a more general level in order to avoid their duplication across the ontology.
18. We select these classifications because they clearly show two alternative, very common approaches for classifying features. The considerations expressed on these models apply to other classifications as well, e.g. Poldermann and Horváth (1996) vs. van Holland and Bronsvoort (2000).
19. Recall that both labels 'form feature' and 'shape feature' can be used in an OWL ontology.
20. For the sake of the example, we assume that Figure 7 represents physical objects. Their information objects counterparts can be easily introduced.
21. The example is formally represented in the available OWL file of the FPRO ontology.

Acknowledgments

Most of the research leading to this publication was done during my fellowship at the ISTC-CNR Laboratory for Applied Ontology. I wish to thank the team of the Laboratory for their support.

Disclosure statement

No potential conflict of interest was reported by the author.

ORCID

Emilio M. Sanfilippo http://orcid.org/0000-0003-2511-2853

References

Baader, F., D. Calvanese, D. L. McGuinnes, D. Nardi, and P. F. Patel-Schneider, eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge: Cambridge University Press.

Barbau, R., S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, and R. D. Sriram. 2012. "OntoSTEP: Enriching Product Model Data Using Ontologies." *Computer-Aided Design* 44 (6): 575–590. doi:10.1016/j.cad.2012.01.008.

Borgo, S., M. Franssen, P. Garbacz, Y. Kitamura, R. Mizoguchi, and P. E. Vermaas. 2014. "Technical Artifacts: An Integrated Perspective." *Applied Ontology Journal* 9 (3–4): 217–235.

Borgo, S., and P. Leitao. 2007. "Foundations for a Core Ontology of Manufacturing." In *Ontologies. A Handbook of Principles, Concepts and Applications in Information Systems*, edited by R. Sharman, R. Kishore, and R. Ramesh, 751–775. New York: Springer US.

Borgo, S., and C. Masolo. 2010. "Full Mereogeometries." *The Review of Symbolic Logic* 3 (4): 521–567. doi:10.1017/S1755020310000110.

Borgo, S., and C. Masolo. 2013. "Foundational Choices in DOLCE." In *Handbook on Ontologies*, edited by S. Staab and R. Studer. Berlin, Heidelberg: Springer Science & Business Media.

Bronsvoort, W. F., and F. W. Jansen. 1993. "Feature Modelling and Conversion. Key Concepts to Concurrent Engineering." *Computers in Industry* 21 (1): 61–86. doi:10.1016/0166-3615(93)90045-3.

Brunetti, G. 2003. "Feature-Based Virtual Engineering." In *Feature based product life-cycle modelling. Conference on Feature Modelling in Advanced Design for the Life Cycle Systems (FEATS), June 12–14,2001, Valenciennes, France*, Springer Science Business Media New York.

Brunetti, G., and S. Grimm. 2005. "Feature Ontologies for the Explicit Representation of Shape Semantics." *International Journal of Computer Applications in Technology* 23 (2): 192–202. doi:10.1504/IJCAT.2005.006481.

Casati, R., and A. Varzi. 1999. *Parts and Places: The Structures of Spatial Representation*. USA: MIT Press.

Case, K., and W. A. Wan Harun. 2000. "Feature-Based Representation for Manufacturing Planning." *International Journal of Production Research* 38 (17): 4285–4300. doi:10.1080/00207540050205091.

Chandrasegaran, S. K., K. Ramani, R. D. Sriram, I. Horváth, A. Bernard, R. F. Harik, and W. Gao. 2013. "The Evolution, Challenges, and Future of Knowledge Representation in Product Design Systems." *Computer-Aided Design* 45 (2): 204–228. doi:10.1016/j.cad.2012.08.006.

Chungoora, N., O. Canciglieri, and R. I. M. Young. 2010. "Towards Expressive Ontology-Based Approaches to Manufacturing Knowledge Representation and Sharing." *International Journal of Computer Integrated Manufacturing* 23 (12): 1059–1070. doi:10.1080/0951192X.2010.518976.

Crofts, N., M. Doerr, T. Gill, S. Stead, and M. Stiff. 2008. "Definition of the CIDOC Conceptual Reference Model." In *ICOM/CIDOC Documentation Standards Group*, 5. http://www.cidoc-crm.org/sites/default/files/cidoc_crm_version_5.0.4.pdf

Efthymiou, K., K. Sipsas, D. Mourtzis, and G. Chryssolouris. 2015. "On Knowledge Reuse for Manufacturing Systems Design and Planning: A Semantic Technology Approach." *CIRP Journal of Manufacturing Science and Technology* 8: 1–11. doi:10.1016/j.cirpj.2014.10.006.

El Kadiri, S., and D. Kiritsis. 2015. "Ontologies in the Context of Product Lifecycle Management: State of the Art Literature Review." *International Journal of Production Research* 53 (18): 5657–5668. doi:10.1080/00207543.2015.1052155.

Fenves, S. J., S. Foufou, C. Bock, and R. D. Sriram. 2008. "CPM: A Core Model for Product Data." *Journal of Computing and Information Science in Engineering* 8: 1. doi:10.1115/1.2830842.

Fontana, M., F. Giannini, and M. Meirana. 1999. "A Free Form Feature Taxonomy." *Computer Graphics Forum* 18 (3): 107–118. doi:10.1111/cgf.1999.18.issue-3.

Guarino, N., S. Borgo, and C. Masolo. 1997. "Logical Modelling of Product Knowledge: Towards a Well-Founded Semantics for STEP." In *Proceedings of European Conference on Product Data Technology*, 183–190. Citeseer.

Guarino, N., D. Oberle, and S. Staab. 2009. "What Is an Ontology?" In Handbook on Ontologies, edited by S. Staab and R. Studer, 1–17. Berlin, Heidelberg: Springer Science & Business Media.

Guarino, N., and C. A. Welty. 2009. "An Overview of OntoClean." In *Handbook on Ontologies*, edited by S. Staab and R. Studer, 201–220. Berlin, HeidelbergSpringer Science & Business Media.

Gupta, R. K., and B. Gurumoorthy. 2013. "Unified Taxonomy for Reference Ontology of Shape Features in Product Model." In *IFIP International Conference on Product Lifecycle Management*, 295–307. Springer.

Han, J. H. 1996. *Survey of Feature Research*. Technical Report IRIS-96–346. Institute for Robotics and Intelligent Systems, USC,USA.

Han, J., M. Pratt, and W. C. Regli. 2000. "Manufacturing Feature Recognition from Solid Models: A Status Report." *IEEE Transactions on Robotics and Automation* 16 (6): 782–796. doi:10.1109/70.897789.

Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. 2004. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML." *World Wide Web Consortium* 21. https://www.w3.org/Submission/SWRL/

ISO 10303 . 2006. *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 224, Mechanical Product Definition for Process Planning Using Machining Features*. ISO:Geneve.

Lu, Y., K. C. Morris, and S. Frechette. 2016. "Current Standards Landscape for Smart Manufacturing Systems." *National Institute of Standards and Technology (NISTIR)* 8107: 22–28.

Ma, Y.-S., G. Chen, and G. Thimm. 2008. "Paradigm Shift: Unified and Associative Feature-Based Concurrent and Collaborative Engineering." *Journal of Intelligent Manufacturing* 19 (6): 625–641. doi:10.1007/s10845-008-0128-y.

Mourtzis, D., and M. Doukas. 2012. "A Web-Based Platform for Customer Integration in the Decentralised Manufacturing of Personalised Products." *Procedia CIRP* 3: 209–214. doi:10.1016/j.procir.2012.07.037.

Nasr, E. A., and A. K. Kamrani. 2007. *Computer-Based Design and Manufacturing: An Information-Based Approach*. New York: Springer-Verlag.

Papakostas, N., G. Pintzos, M. Matsas, and G. Chryssolouris. 2014. "Knowledge-Enabled Design of Cooperating Robots Assembly Cells." *Procedia CIRP* 23: 165–170. doi:10.1016/j.procir.2014.10.092.

Pauwels, P., and W. Terkaj. 2016. "EXPRESS to OWL for Construction Industry: Towards a Recommendable and Usable IfcOWL Ontology." *Automation in Construction* 63: 100–133. doi:10.1016/j.autcon.2015.12.003.

Poldermann, B., and I. Horváth. 1996. "Surface Design Based on Parametrized Surface Features." In *Proc. Int. Symposium on Tools and Methods for Concurrent Engineering*, Institute of Machine Design, *Budapest*, 432–446.

Rentzos, L., K. Smparounis, D. Mavrikios, and G. Chryssolouris. 2012. "An Ontology for Classifying Advanced Visualization Infrastructures." In *14th International Conference on Modern Information Technology in the Innovation Processes of Industrial* Enterprises *(MITIP)*.

Rijgersberg, H., M. Van Assem, and J. Top. 2013. "Ontology of Units of Measure and Related Concepts." *Semantic Web* 4 (1): 3–13.

Romero, F., P. Rosado, and G. M. Bruscas. 2015. "Application Feature Model for Geometrical Specification of Assemblies." *Procedia Engineering* 132: 1128–1135. doi:10.1016/j.proeng.2015.12.605.

Rossignac, J. R. 1990. "Issues on Feature-Based Editing and Interrogation of Solid Models." *Computers & Graphics* 14 (2): 149–172. doi:10.1016/0097-8493(90)90029-W.

Sakurai, H., and C.-W. Chin. 1994. "Definition and Recognition of Volume Features for Process Planning." In *Manufacturing Research and*

Technology, Edited by J. J. Shah, M. Mäntylä, D. S. Nau, Vol. 20, 65–80. Elsevier. https://www.sciencedirect.com/bookseries/manufacturing-research-and-technology/vol/20/suppl/C

Sanfilippo, E. M., and S. Borgo. 2016. "What are Features? an Ontology-Based Review of the Literature." *Computer-Aided Design* 80: 9–18. doi:10.1016/j.cad.2016.07.001.

Sanfilippo, E. M., F. Mandorli, C. Masolo, and M. Ragni. 2017. "The Interplay between Shape and Feature Representation." In *Proceedings of the Joint Ontology Workshops*, Vol. 2050. Ceur workshop proceedings.

Sanfilippo, E. M., C. Masolo, S. Borgo, and D. Porello. 2016. "Features and Components in Product Models." In *Formal Ontology in Information Systems. Proceedings of the 9th International Conference (FOIS)*, edited by R. Ferrario and W. Kuhn, Vol. 283, 227–240. doi: 10.1098/rspb.2016.0343

Schulte, M., C. Weber, and R. Stark. 1993. "Functional Features for Design in Mechanical Engineering." *Computers in Industry* 23: 15–24. doi:10.1016/0166-3615(93)90111-D.

Shah, J. J., and M. Mäntylä. 1995. *Parametric and Feature-Based CAD/CAM. Concepts, Techniques, Applications*. Canada: John Wiley and Sons.

Solano, L., F. Romero, and P. Rosado. 2016. "An Ontology for Integrated Machining and Inspection Process Planning Focusing on Resource Capabilities." *International Journal of Computer Integrated Manufacturing* 29 (1): 1–15.

Štorga, M., M. M. Andreasen, and D. Marjanovic. 2010. "The Design Ontology: Foundation for the Design Knowledge Exchange and Management." *Journal of Engineering Design* 21 (4): 427–454. doi:10.1080/09544820802322557.

Stroud, I., and H. Nagy. 2011. *Solid Modelling and CAD Systems: How to Survive a CAD System*. London: Springer Science & Business Media.

Tang, S.-H., G. Chen, and Y. Ma. 2013. "Fundamental Concepts of Generic Features." In *Semantic Modeling and Interoperability in Product and Process Engineering*, edited by Y.-S. Ma, 89–115, London: Springer -Verlag.

Tessier, S., and T. Wang. 2013. "Ontology-Based Feature Mapping and Verification between CAD Systems." *Advanced Engineering Informatics* 27: 76–92. doi:10.1016/j.aei.2012.11.008.

Uddin, M., and Y.-S. Ma. 2015. "A Feature-Based Engineering Methodology for Cyclic Modeling and Analysis Processes in Plastic Product Development." *Computer-Aided Design and Applications* 12 (6): 1–12.

Usman, Z., R. I. M. Young, N. Chungoora, C. Palmer, K. Case, and J. Harding. 2013. "Towards a Formal Manufacturing Reference Ontology." *International Journal of Production Research* 51 (22): 6553–6572. doi:10.1080/00207543.2013.801570.

van den Berg, E., W. F. Bronsvoort, and J. S. M. Vergeest. 2002. "Freeform Feature Modeling: Concepts and Prospects." *Computers in Industry* 49 (2): 217–233.

van Holland, W., and W. F. Bronsvoort. 2000. "Assembly Features in Modeling and Planning." *Robotics and Computer-Integrated Manufacturing* 16 (4): 277–294. doi:10.1016/S0736-5845(00)00014-4.

W3C. 2009. "OWL 2 Web Ontology Language Document Overview." In *World Wide Web Consortium*. https://www.w3.org/TR/owl2-overview/

Weichhart, G., A. Molina, D. Chen, L. E. Whitman, and F. Vernadat. 2016. "Challenges and Current Developments for Sensing, Smart and Sustainable Enterprise Systems." *Computers in Industry* 79: 34–46. doi:10.1016/j.compind.2015.07.002.

Wingård, L. 1991. "Introducing form features in product models: a step towards CAD/CAM with engineering terminology." PhD diss., Dep. of Manufacturing Systems, Royal Institute of Technology, Stockholm.

Yusuf, Y., and Y. Ma. 2016. "Design of a Simulation Tool for Steam Assisted Gravity Drainage: Based on the Concept of Unified Feature Modeling Scheme." In *Proceedings of TMCE*, edited by I. Horváth, J.-P. Pernot, Z. Rusák, 647–658. Vol. 2016.

Zhang, D. J., F. Z. He, S. H. Han, and X. X. Li. 2016. "Quantitative Optimization of Interoperability during Feature-Based Data Exchange." *Integrated Computer-Aided Engineering* 23 (1): 31–50. doi:10.3233/ICA-150499.

Zhang, Y., X. Luo, B. Zhang, and S. Zhang. 2017. "Semantic Approach to the Automatic Recognition of Machining Features." *The International Journal of Advanced Manufacturing Technology* 89 (1–4): 417–437. doi:10.1007/s00170-016-9056-8.

Zhou, X., Y. Qiu, G. Hua, H. Wang, and X. Ruan. 2007. "A Feasible Approach to the Integration of CAD and CAPP." *Computer-Aided Design* 39 (4): 324–338. doi:10.1016/j.cad.2007.01.005.

Appendix: Representing and reasoning over products and features

The LUPO and FPRO ontologies presented throughout the paper are now used to represent the products in Figure 7,[20] and to reason over them. The figure shows an assembled product, named *p*3 , which has *p*1 and *p*2 as components. Additionally, *p*1 and *p*2 have void P-features *f*1 and *f*2 , respectively. The composition of the last two features generates *f*3 , which is the compound P-feature of *p*3 . For the sake of simplicity, we do not specify what types of features *f*1 , *f*2 , and *f*3 are, e.g. whether they are form or manufacturing features; holes, slots, or pockets, etc. Also, we do not represent qualities (e.g. shape, weight, dimensions, etc.), nor the compliance of the objects at stake with corresponding design (information) objects. However, LUPO and FPRO provide the axiomatic knowledge to easily introduce this information.[21]

From the formulas below (f7) – (f10), the *explicit* information stated in the A-box of the ontology is that: (i) *p*1 and *p*2 are (different) components of *p*3 , and they are both single material products; (ii) *p*1 and *p*2 have features *f*1 and *f*2 , respectively; (iii) *f*1 is a void P-feature; (iv) *f*1 and *f*2 are both component of *f*3 . Note that we do not know what type of feature *f*2 is, nor we have information about *p*3 and *f*3.

F7 Individual : *p*1
  Types:
    SingleMaterialProduct
  Facts:
    componentOf *p*3
    hasFeature *f*1
  DifferentFrom:
    p2
(*p*1 is a single material product that is component of *p*3 , has feature *f*1, and is different from *p*2)

f8 Individual : *p*2
  Types:
    SingleMaterialProduct
  Facts:
    componentOf *p*3
    hasFeature *f*2
(*p*2 is a single material product that is component of *p*3 and has feature *f*2)

f9 Individual : *f*1
  Types:
    VoidPFeature
  Facts:
    componentOf *f*3
  DifferentFrom:
    f2
(*f*1 is a void p-feature that is component of *f*3 and is different from *f*2)

f10 Individual : *f*2
  Facts:
    componentOf *f*3
(*f*2 is component of *f*3)

By reasoning over the knowledge base, we automatically infer that:

(1) *f*1 is feature of *p*3. This is mainly inferred because of (R3), hence since *f*1 is feature of *p*1, which is component of *p*3, then *f*1 is feature of *p*3, too;
(2) *f*2 is feature of *p*3, given that it is feature of *p*2, which is component of *p*3 (R 3). Also, *f*2 is classified as an instance of VoidPFeature. This because both *f*1 and *f*2 are declared as components of *f*3, and *f*1 is a void P-feature. We know from axioms like (Ax 6), (Ax7), and (R1), among others, that parthood (hence componenthood) is restricted between either material or immaterial objects. Since *f*1 is a void P-feature (thus an immaterial object) that is component of *f*3, the latter is a void P-feature, too, thus all its (feature) components must be void P-features;
(3) *f*3 is classified as an instance of both VoidPFeature and CompoundPFeature. We saw the explanation of the first inference above; the second one is due to the fact that *f*3 has both *f*1 and *f*2 as (feature) components, see (Def 10) and (Def 11);

(4) *p*3 is classified as an instance of AssembledMaterialProduct. Recall, indeed, that only assembled products have components, see (Def 4) and (Def 5);
(5) *p*3 has (compound) feature *f*3. This is inferred mainly by (R 4), which allows to aggregate features to products by reasoning over the relationships of featureOf and componentOf.

Note that despite both features *f*1 and *f*2 are represented as P-features of products *p*1 and *p*2, respectively, the reasoner does not classify them as *parts* of the products. Compare this situation with formulas (f 11) – (f 12) below representing the single product *p*4 with material feature *f*4. The formulas may be satisfied by a block with a bump, see Figure 8.

f11 Individual : *p*4
  Types:
    SingleMaterialProduct
  Facts:
    hasFeature *f*4
(*p*4 is a single material product with feature *f*4)

f12 Individual : *f*4
  Types:
    MaterialPFeature
(*f*4 is a material p-feature)

By calling the reasoner, *f*4 is classified as (proper) part of *p*4. This is due to (R 1), which establishes a correspondence between the relations featureOf and properPartOf in the case of material features. Also, note that, despite componentOf is subsumed by properPartOf, *f*4 is not classified as component of *p*4, since products can have only other products as components. As said in Section 4, by distinguishing between properPartOf and componentOf we can represent the parts of a product without committing to its components.