UNIVERSITÀ DEGLI STUDI DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE

ICT International Doctoral School

Ontological foundations for feature-based product modelling

Emilio M. Sanfilippo

Advisor

Stefano Borgo

Laboratory for Applied Ontology (ISTC-CNR)

Co-Advisor

Nicola Guarino

Laboratory for Applied Ontology (ISTC-CNR)

January 2017

Contents

Acknowledgements **4**

Abstract **5**

1 Introduction **6**
1.1 Research problem 6
1.2 Research contributions 9
1.3 Research methodology 10
1.4 Thesis overview 14

2 State of the art review **15**
2.1 Design 15
2.2 Product information modelling 17
2.2.1 Feature-based modelling: An overview 17
2.2.2 Features in standards 22
2.2.3 Features in information models 24
2.3 Open problems 29

3 DOLCE **32**
3.1 Introduction 32
3.2 Particulars 33
3.3 Mereology 34
3.4 Qualities 35
3.5 Concepts and roles 39
3.6 Objects and events 40
3.7 Material and immaterial objects 42

4 Technical products and product types **44**
4.1 Artefacts 44
4.1.1 Products 45
4.1.2 Compliance 46
4.1.3 Taxonomy of artefactual notions 48
4.2 Design properties 49
4.2.1 Product types as concepts 52

4.3 Comparison with the literature 54

5 Engineering features **60**

5.1 What do engineering features represent? 60

5.2 Feature types and physical features 63

5.3 Ontological features: An informal overview 65

5.4 Components 67

5.5 Form features 69

5.6 Comparison with the literature 73

6 Formal representation **75**

6.1 Ontological features 76

6.2 Design properties 79

6.3 Product types and technical products 83

6.4 Engineering features 85

6.5 Form features: Three modelling views 87

6.5.1 First view: Form features for shape qualities 87

6.5.2 Second view: Form features for shaped parts 88

6.5.3 Third view: Form features for shaped immaterial objects 89

6.5.4 Discussion and comparison 90

6.5.5 An integrated perspective 92

6.6 Remarks 95

7 Case studies **98**

7.1 Form features representation: An example 98

7.2 Towards a feature-based meta-model 106

7.2.1 Unified Feature Model 106

7.2.2 Feature-based Meta-Model 108

7.2.3 Critical remarks 110

7.2.4 Ontology-based restructuring of the meta-models . . . 112

8 Conclusion **115**

8.1 Limits of the approach 117

8.2 Future work 118

Appendices **123**

A The missing and replaceable artefact **124**

B Glossary of terms **129**

Bibliography **131**

Simply as a mental exercise, without any assertion that it is true, let me indicate a possible line of thought. It is, I admit, mere imagination; but how often is imagination the mother of truth?

Sherlock Holmes

There are more ontological options than kinds of coffee.

Huw Price

Acknowledgements

I could not have written this thesis without the scientific and personal support of different people. I wish to thank Stefano Borgo, Claudio Masolo, Nicola Guarino and Daniele Porello for their patient and willingness in sharing their knowledge and experience. I am particularly thankful to Stefano for his precious mentorship and to Claudio for the uncountable discussions. I wish also to thank Roberta Ferrario, Laure Vieu, Tiago Sales, Francesco Setti, Nicolas Troquard, Isabella Distinto, Emanuele Bottazzi, Chiara Bassetti, as well as colleagues at the Institute of Industrial Technologies and Automation (ITIA-CNR), Walter Terkaj, Alexandra Sojic and Marco Sacco for their support.

I am grateful to Michael Grüninger and Bob Young for their reviews. I wish to thank the latter, together with Esmond Urwin and Clair Palmer, also for the visiting period I spent at Loughborough University. The Italian Association for Artificial Intelligence (AIXIA) supported me with a grant during the visiting. I am thankful for that.

Ulf Schwarz, Mathias Brochhausen and Luc Schneider introduced me to the adventure of Applied Ontology. I wish to thank all of them; the thesis is part of what they started.

The path that brought me to the ICT Doctoral School in Trento started with Giovanni Camardi, who has been since many years a key figure in my life. I am grateful to him for his precious advices.

The thesis is dedicated to Barbara, to her curiosity, music and eyes.

Abstract

Product modelling and design data management are knowledge intensive tasks carried out by means of computer systems that allow experts to represent, share and possibly integrate disparate quantitative and qualitative models. Feature-based modelling is the leading approach for computer-based product representation. Traditionally, features are represented via object-oriented methods for information management. However, the development of advanced computer systems calls for ontologies, which provide formal means to share and integrate heterogeneous knowledge in a way that is transparent to experts and is accessible to automated reasoners.

Despite their relevance, the development of feature-based ontologies has not been supported by a systematic treatment of the notions at hand, nor by an exploration of the various alternatives by which features can be interpreted and modelled. At the state of art, it remains unclear what features are and how different modelling perspectives can be coherently represented within a larger theory for product knowledge representation.

The work presented in this thesis strives from the need of providing ontological foundations for product knowledge management; in particular, it provides an ontological characterisation of engineering features on the grounds of foundational theories of ontology engineering.

Inasmuch as feature-based models are employed for disparate application tasks, on the one side we develop a high-level understanding of features by distinguishing the domain entities that are modelled in their terms. In this manner, we aim at an ontology that is not strictly committed to specific application requirements but it is broadly applicable to product knowledge representation. On the other hand, we focus on the representation of (some aspects of) the physical layouts of material products. The notion of form feature plays a relevant role in this context and attention will be paid in the different ways in which it can be understood and represented according to engineering conceptualisations.

Chapter 1

Introduction

1.1 Research problem

Engineering design concerns the ideation and development of products. It involves disparate communities of experts, from market analysts and business experts, to engineers, designers and technicians, among others, each community speaking with its own terminology and looking at the product to be developed from its own perspective [PDS05, MCT08]. A seller, for instance, may be interested in the novelties of a product with respect to similar devices available in the market, whereas a technician responsible for product realisation may look at a product in terms of the materials and tools necessary for its production.

Nowadays, Computer Technologies (CAx) like Computer-Aided Design (CAD) or Computer Aided Process Planning (CAPP) systems are intensively used to support different product development phases [DWGB04]. CAx systems allow not only for the representation and management of quantitative (geometrical, topological, algebraic) product data, but also for the embedding of qualitative constraints concerning, e.g., design intents, functional descriptions and business strategies into product models [CMS07, JGAB09]. The exploitation of these systems call for expressive and integrated data models and approaches that are able to capture and manage relevant product information. In particular, experts have been looking for modelling approaches that are able to abstract from single applications and scenarios to explicitly represent the semantics of the employed terms. This raises the issue of how to specify technical knowledge in a way that reflects experts' understanding of domain entities,[1] is semantically transparent to third parties, is possibly tractable by automated reasoners and is able to guarantee inter-systems interoperability [EKGT$^+$16].[2]

[1]With *domain entities* we refer to the things experts or companies talk about in their everyday practises [Bor14].

[2]According to [Weg96], "[i]nteroperability is the ability of two or more software [...]

Feature-based modelling is the leading approach for computer-based product development and information management, since it enables the specification of qualitative data into pure geometric models [Ma13, RRB15]. For example, a feature-based model can be used to represent both product geometry and the operations for its manufacturing, allowing in this way for the integrated representation of design and manufacturing data [DGG$^+$07]. Accordingly, 'feature' – the key notion in *feature*-based approaches – refers to modelling elements that are used to codify experts' knowledge in a way that integrates multiple expertise perspectives.

Traditionally, features are represented in CAx applications as lists of values and rules. At the level of data management, they are treated as classes of entities with common attributes classified into taxonomies, from the most general to the most specific classes [TCM13]. However, the development of knowledge-based CAx systems calls for computational models that do not limit to the taxonomical representation of the classes at hand. They rather require the exploitation of theories, whose modelling elements (classes, relations, axioms) reflect technical knowledge in a systematic and unambiguous manner. These theories are called *ontologies* (see [Sect. 1.3] below).

Although the great efforts towards the formal treatment of product knowledge, the development and exploitation of information models have not been systematically carried out. More specifically, ontologies grounded on feature-based approaches suffer from different drawbacks.

First, they either under-specify the meaning of the represented terms (*semantic underspecification*), or make tacit and often unclear assumptions about them, therefore about the domain entities to which they refer (*opaque ontological commitments*).[3] For instance, the term 'product', whose meaning should be clearly represented given its relevance for *product* knowledge representation, is generally represented as something that has a name, an identifier or production date [San15]. Informally, however, experts assume much more about what products are, e.g., whether they are the result of human labour, or whether they are necessarily constituted by materials.[4] Semantic underspecification and the opacity in the ontological commitments of current information models raise the problems of data management and semantic interoperability; computer systems can indeed hardly exchange data if the semantics behind is unclear.

to cooperate despite differences in language, interface, and execution platforms", where 'cooperation' has to be understood in terms of the systems ability to share (and possibly) integrate their data while keeping the intended semantics.

[3]In the literature about ontology (covering both philosophy and computer science), the expression 'ontological commitment' (or 'existential commitment') refers to the entities that are assumed to exist in a domain according to a theory [Gua98]. For example, the ontological commitment of biology is about living organisms, cells, etc.

[4]We shall dig into the different ways of understanding products in [Ch. 4]. For the time being, we note that products may not be material objects; services and software are two examples.

Second, information models have not been developed within broader ontological theories, which distinguish in a clear manner between different domain entities, e.g., products, components and their features. To be more precise, current works do distinguish between various classes of entities, but the motivations behind these distinctions are not motivated. In [RRB15], for example, some features are represented as components, but the rationale for such a modelling approach is unclear (see [Sect. 7.2] in [Ch. 7] for details).

Third, focusing on the very notion of feature, which is at the core of our study, after more than 30 years of feature-based approaches, it is neither clear what a feature is meant to represent, nor how it distinguishes from non-feature entities. For instance, a feature is sometimes treated as a product's quality and sometimes as a physical object that can be present in space and time only if attached to some non-feature entity but without reducing to qualities. However, neither the rationales behind these two views are motivated, nor it is made explicit when one approach should be preferred over the other. More generally, if on the one hand features are extensively employed for disparate modelling scenarios and applications (see [Ch. 2]), on the other hand there is no systematic treatment of the various ways in which features are understood. As stressed in [EKGT$^+$16, SB16], before implementing software solutions or algorithmic procedures to support product development, the terms relevant for the applications have to be identified and understood.

In a more general perspective, the development of ontologies for product knowledge representation has not been supported by principled methodologies for the analysis of domain knowledge and its formal specification. The emblematic example is the current effort towards the codification of data modelling standards in logical languages [BVLDV09, KBF$^+$09, PT16]. On the one side, this will facilitate automatic reasoning procedures, as well as the exploitation of Semantic Web technologies [HKR09] for product design. On the other hand, formal logic is notoriously a tool that for knowledge representation purposes needs to be supported by domain theories (ontologies), which organise the represented knowledge in a principled manner, avoiding rules of thumb and *ad hoc* modelling strategies. Codifying standards in formal logic does not necessarily clarify the intended meanings of the represented notions, nor does it support by itself, e.g., smoothness and meaningful data sharing, since it does not guarantee the coherent treatment of technical knowledge with respect to experts' conceptualisations.

1.2 Research contributions

The overall purpose of this research project is to contribute to the development and application of ontologies for product knowledge representation. More specifically, our aim is to explore and provide the foundational basis of feature-based models in product design on the grounds of foundational theories of ontology engineering (see [Sect 1.3]). *Foundational basis* means that notions relevant for product modelling are characterised in terms of a high-level ontological perspective by which their meaning is specified within a broader understanding of the domain entities that are assumed to exist according to experts' background knowledge. Additionally, our theory – although grounded on engineering practise and knowledge – is not directly based on application concerns, in the sense that it abstracts from specific modelling scenarios and is therefore exploitable (at least in principle) in disparate applications. As we will see, this means that in order for the proposed ontology to be used in real-world modelling scenarios, it needs to be integrated with knowledge suitable for specific application requirements.

The following research goals are at the core of the proposed work:

Goal 1: Development of a high-level ontological framework for product knowledge representation. As said, the ontological modelling of features has to be contextualised within a broader ontological interpretation of engineering knowledge. In particular, given the relevance of product models for product development and data sharing, we shall propose an approach that maintains a sharp distinction between (generally speaking) physical entities (e.g., products, components) and (what we call) the *design properties* that they are required to satisfy. This distinction will lay at the ground of our approach and will be also adopted to represent features;

Goal 2: Identification of the domain entities represented in feature approaches. By reviewing the state of the art relevant to our work, we will identify the basic notions to be systematically analysed to support the development of feature-based product models. In particular, as we shall see throughout the thesis, form features are core elements in feature modelling. We will therefore look at how they are conceptualised across the literature to provide their ontological foundations;

Goal 3: Ontological analysis and formal representation of the concepts previously identified. As stated earlier, we shall rely for this purpose on a methodology based on the use of a foundational ontology. Since feature-like notions are heterogeneously understood across the literature, we do not in first instance prescribe a monolithic ontological characterisation, which may fail in meeting both experts' views and application concerns. Rather, we firstly look at the different conceptualisations proposed across the literature; we then evaluate their pros and cons on the basis of some recurrent modelling scenarios, and finally propose how different perspectives can be integrated in an unified view. Also, our work does not attempt a formal representation of features that can convey geometric or topological information. These types of information can be indeed well handled in current modelling approaches. As said, our aim is to provide a formal, ontological characterisation of features. Once this has been done, the ontology can be enriched with more specific geometric and/or topological constraints.

Since product design and product knowledge representation are broad fields of research, we narrow the scope of our work to material products. Hence, services and software are out of the scope of this thesis.

1.3 Research methodology

The analysis of product knowledge and its formal representation is based on the foundational and formal methods of ontology engineering [SS13, GW00]. This choice relies on the fact that, as we previously saw, ontologies are broadly exploited in product design for disparate applications [EKK15].

Traditionally, Ontology is one of the oldest branch of philosophy concerned with the question of what there exists [Var10].[5] Its core aim is to define *a system of inter-related categories of individual entities* (namely an *ontology* or *ontological theory*), whose existence is either given for granted in reality, or it is assumed in some background theory or common sense worldview. Differently from other research enterprises such as natural sciences, ontologists typically abstract from specific cases and provide very general yet fundamental categories of what exists, 'fundamental' in the sense that everything else is grounded on these categories. A biologist, for example, may ask what is the difference in shape and size of different animal cells. On the other hand, an ontologist would ask whether cells, shapes and sizes can be classified by the same category in virtue of the properties they satisfy. As another example, a mathematician may calculate complex numerical expressions via the rules of calculus, whereas an ontologist may wonder about what a number is, e.g., whether numbers exist on their own independently of human mind, or whether they are some sort of human-made constructs. From this perspective, Ontology is sometimes understood as providing a rigorous methodology for conceptual analysis, namely to investigate and compare different systems of thinking and to lighten their (often hidden) existential assumptions. For the purposes of our work, *formal Ontology*

[5]We borrow from [GG95] the use of 'Ontology' (with the capital 'o') to refer to the philosophical discipline.

plays a key role, because it is concerned with the axiomatic treatment of very general categories, which are (at least in principle) independent of any specific domain of reality or theory thereof. Formal Ontology distinguishes from *material Ontology*, which analyses specific categories concerning, e.g., physics or biology [Var10, Coc91].

The core ideas of what is nowadays called *ontology engineering* or *applied ontology* [GPFLC04, MS08] root back at the beginnings of Artificial Intelligence (AI), although the term 'ontology' made its first explicit appearance in computer science in a work related to the foundations of data modelling (see [Gui05, Ch.3]). In a seminal paper, McCarty and Hayes [MH69] claimed that an agent has to be provided with a *model of the world*, an ontology indeed, in order to act intelligently. The first definition of ontology in computer science was given later by Gruber [Gru93] in the context of knowledge-based systems; it was then systematically revised by Guarino [Gua95, Gua98], who bridged AI research and philosophical Ontology. Accordingly, an ontology is *a formal and explicit specification of a shared conceptualisation* [GOS13, SBF98], namely a theory whose interpretations reflect someone's knowledge about an application domain (e.g, engineering design or geospatial science). A *conceptualisation* is a world-view about the entities that are assumed to exist in the domain at stake, their properties and the relationships they share; a conceptualisation has to be *shared* in order to capture consensual knowledge. An ontology is *explicit* because it has to provide a transparent representation of the meaning of its terms so that its vocabulary can be accessible without the risk of missing relevant information. An ontology is *formal* because represented in a language with formal semantics, which provides a rigorous mechanism to control the interpretations of the ontology.[6]

More precisely, given a domain D and a language L with vocabulary V, an ontology O of D is a set of V-propositions formulated in L by which (some of) the entities in D and (some of) their properties are represented [HH06]. If L is a first-order language (FOL), then O is a first-order theory, whose interpretations (should) correspond to a shared world view.[7] As Guarino [Gua98] pointed out, an ontology can only approximately represent a domain, because it abstracts those aspects that are relevant for some application from the perspective of a community.

[6]Ontologies in computer science are not necessarily formalised in computer tractable languages and are sometimes expressed in natural language, especially when employed for inter-human communication rather than computer implementations [SW01]. The label 'computational ontology' is often used with reference to computer tractable ontologies such as ontologies in Description Logics [BCM$^+$03].

[7]Given a first-order vocabulary V with function, relation and constant symbols, and a V-structure (call it S), namely a non-empty set U of individuals along with an interpretation function I that associates individuals of U to constants in V, functions from U^n to U to each n-ary function in V, and subsets of U^n to n-ary relations in V, a first-order theory of S (denoted as $Th(S)$) is a set of V-propositions which hold in S [Hed04].

Although ontologies are currently employed for various application tasks, their purposes are mainly twofold: firstly, to provide theories for knowledge representation. From this perspective, there is a strong link between ontology engineering and *qualitative knowledge representation and reasoning* [Sto98, Gal00]. They both provide, indeed, theories that do not aim at the same preciseness given by quantitative, mathematical models but reflect everyday human reasoning. Secondly, an ontology can be used to represent the meaning of a vocabulary. An ontology is thus a way to convey the semantics of terms employed within a community [GOS13, GM05]. This facilitates communication, information retrieval, database management and data sharing, among others, because the meaning of the employed vocabulary is constrained to rule out undesired interpretations.

Since ontologies are models that describe (portions of) reality for practical needs, a comment is due on the distinction between ontologies and other models developed in computer science. The distinction is rather challenging and despite the scientific debate [Gua94, SC10, HS12], no standardised view is available. We assume that an ontology, differently from other models, has to be explicit about the high-level distinctions between the entities that exist in the represented domain. For example, if an ontology only recognises entities located in both space and time, it has to avoid altogether abstract entities like numbers.[8] Also, an ontology should avoid the use of mixed categories that bring together incompatible properties [Bor14], e.g., 'being an idea' *and* 'being a physical object'. If we assume that instances of the former property are abstract entities, differently from instances of the latter, then the properties are incompatible because an entity cannot both have and lack spatio-temporal location.[9] More importantly, an ontology has to specify the identity conditions of the domain entities it deals with, namely the properties that an individual has to necessarily satisfy. Notoriously, the definition of these properties is a complex task [GW00]. Without entering into the philosophical debate, from an engineering perspective to model an entity's identity means to represent the properties whose change involves a change in the entity's nature. Consider, e.g., John, who is both a person and a student. We may think that John can stop being a student while remaining the same, whereas it cannot stop being a person without changing in some radical way. From this perspective, 'being a person' is a necessary property that determines John's identity. Similarly, when one deals with engineering knowledge, attention has to be paid in individuating the properties that the represented entities have to necessarily satisfy. For example, a certain product may be designed with a specific functionality so that if the product cannot perform the functionality, it cannot be considered as a product at

[8]For the sake of the example, let us assume that numbers are abstract entities, i.e., entities that are neither in space nor in time.

[9]See [ABAS16] for a violation of this constraint.

all. Accordingly, 'having a designed functionality' may be seen as a property that determines products' identities, differently from a property like 'having an identifier'.

As claimed at the beginning of the section, we rely on a foundational approach to ontology engineering. Generally speaking, a foundational ontology is an upper-level ontology that captures the meaning of domain-independent notions [Sch03, Kee11]. Differently from other upper-level ontologies, a *foundational* ontology is richly axiomatised and grounded on theories of philosophical Ontology, artificial intelligence, linguistics and often also cognitive science (e.g., [MBG$^+$03, Gui05]). The idea is to provide a trust source of knowledge to avoid relying on *ad hoc* and scarcely reusable representational approaches. Foundational ontologies are thus strictly related to formal (philosophical) ontologies, because of their focus on the axiomatisation of general categories of entities.

From an application perspective, foundational ontologies are exploitable in computer systems only when specialised with specific knowledge and formalised in tractable languages. For this reason, they are commonly adopted as starting frameworks for the development of domain- or application-driven ontologies, or to compare and possibly integrate different ontologies by their alignment under the same upper-level umbrella [AHYC12b]. Recall that different foundational ontologies are currently available; the idea, indeed, is not to rely on a single foundational ontology, but to have a *library* by which different foundational ontologies can be compared in terms of both their existential commitments, representational choices and application scenarios [MBG$^+$03].

1.4 Thesis overview

The thesis is structured as follows. In [Ch. 2] we present the state of the art relevant for our purposes. Given our focus on feature-based product representation, we dig into various interpretations of features found across the engineering literature. We then look at current data modelling resources and by the end of the chapter address the open issues, which we address in the remaining chapters. In [Ch. 3] the DOLCE foundational ontology is introduced as reference theory for the development of our work. As previously said, indeed, one of the main motivation behind our work is to contextualise an ontological representation of features within an overall theory of entities assumed to exist in (common-sense) reality. Also, an ontology for features needs to explain how features relate and distinguish from other entities relevant for product development, products foremost. In [Ch. 4] we thus dig into the analysis of what a product is meant to be in engineering conceptualisations. Furthermore, we address in the same chapter a basic distinction between physical products and the design representations where experts capture the properties that products have to satisfy. As we will better see, this distinction plays a fundamental role and lays at the heart of our approach. In [Ch. 5], the notion of feature is taken into account; the purpose is to identify the main entities modelled in feature-based approaches. A particular emphasis is given to the notion of *form feature*, because of its relevance for product modelling. In [Ch. 6] we present the axiomatisation of the ontology in first-order logic. The formulas are explained and exemplified throughout the chapter. In [Ch. 7] we show how the ontology can be used to model form features in an ontological coherent manner, and how current meta-models for feature representation can be restructured on an ontological basis. We conclude the thesis by addressing future research lines that deserve some attention as a consequence of our work [Ch. 8]. In [Appendix A] we show how our approach can deal with a problem concerning the ontological status of missing and replaceable physical products. In [Appendix B] we summarise the terminology used in our ontology.

Chapter 2

State of the art review

In this chapter we review the literature relevant for our purposes; we therefore focus on the formal treatment of product knowledge and, more specifically, on feature-based product modelling approaches.

Our analysis of the literature is driven by theoretical insights and formal approaches of ontology engineering [SS13, Gua98]. The state of art hereby presented, instead of evaluating technological solutions or algorithmic procedures by which most product models are implemented, focuses on the conceptual models behind these implementations and aims at isolating the foundational assumptions by which features are identified and represented within the context of product knowledge representation.

2.1 Design

Most of the objects we daily handle, if not all of them, have been created on purpose. Pencils, silvers, laptops, washing-machines and cars are only a very few examples of items that have been carefully designed and manufactured for market requests. To put it with the words of Simon [Sim96]: "The world we live in today is much more a man-made, or artificial, world that it is a natural world. Almost every element in our environment shows evidence of human artifice. The temperature in which we spend most of hours is kept artificially at 20 degrees Celsius; the humidity is added to or taken from the air we breathe; and the impurities we inhale are largely produced (and filtered) by man" [ibid., p.2]. Along the same lines, Gareth [Gar15] claims that: "Almost everything we experience has been designed, and not only the myriad of products that we buy. It is possible to talk about the design of services and systems, such as welfare, education and health, often informed by political and economic dogmas but nonetheless containing a belief in their capacities to deliver progress and create a better world" [ibid., p.10]. To put it shortly, "design completely surrounds us" [ibid.].

What is design then? There is no straightforward reply to this question (see [Hor04, Sti90, CB14, PB13, AHC15] among others) and some even doubt about the attainability (and even desirability) of a unified design theory concerning the general principles and practical methodologies of product design [Gal08]. The term 'design' itself is used with different meanings. Flusser [Flu03] provides the following etymological analysis for the term: "In English the word 'design' is both a noun and a verb [...]. As a noun it means – among others – 'intention', 'purpose', 'aim', 'plan' [...]. As a verb (*to design*) it means 'to plan something', 'to simulate', 'to create', 'to sketch', 'to organize' [...]. The term comes from the latin *signum*, which means 'sign' [...]".[1] Flusser's analysis matches with the current uses of 'design' across the specialised literature. The term is indeed associated either to models by which the properties of the products to be developed are described, or to "problem solving process[es]" [Gar15] by which (generally speaking) "technical solutions" to practical needs are created. Interestingly, different authors understand (the activity of) designing as aimed at changing and possibly improving the world around us. Simon [Sim96] claims that "everyone designs who devises courses of action aimed at changing existing situations into preferred ones" [ibid., p.111]. Galle [Gal08] revises Simon's claim in order to constrain designing to the development of artefacts: "Everyone designs who devises courses of action aimed at the production of an artefact". Similarly, according to Gero [Ger90] "[d]esign exists because the world around us does not suit us, and the goal of designers is to change the world through the creation of artifacts" (see also Gareth [Gar15] on the same lines).

A designing activity typically consists of different phases during which expert teams (e.g., designers and manufacturers) interact to produce artefacts that satisfy pre-collected requirements. The so-called *conceptual* and *detail* designing are examples of such phases [UE00, PB13]. In the former experts explore the possible solutions to meet customers' demands, as well as technical requirements (e.g., manufacturing constraints); the solution space is then progressively narrowed until a decision is made about the product to be realised. In the latter phase the identified solution is accurately developed to lead to production. The output of the detail designing phase is a documentation describing the nominal geometry of the product under development along with tolerances, constituting materials and the plan necessary for production, among other information [UE00].

For its very nature designing is collaborative and experts involved in product development continuously interact to share their expertise [Gar15]. For instance, the interaction between designers and manufacturers is fundamental to establish production constraints since the early designing phases.

Nowadays, computer systems are heavily employed to model the entire product lifecycle, from the early product conception to its eventual disposal, to simulate and analyse products' behaviours, to create feasible manufactur-

[1]The translation from the Italian edition is our.

ing plans and (more importantly for our purposes) to share product models within and across organisations. We shall provide a more detailed overview of the application of computer systems in design in the next sections.

2.2 Product information modelling

Computer-based technologies (CAx) are applied for various designing tasks, from functional modelling, to detailed designing, process planning and engineering analysis, among others. From an historical perspective, *Computer-Aided Design* (CAD) systems have been primarily employed for the specification of product geometry but they lack short of enabling the representation of qualitative aspects about design intents [AH15, CRT10]. This stimulated the development of the so-called *feature-based* systems and approaches to allow for the explicit embedding of qualitative knowledge into geometric models [SM95, Xu09]. Nowadays, *knowledge-based* CAx systems support both the quantitative and the qualitative specification of product knowledge, but also the integration of multiple models developed by different working teams, as well as data sharing and reasoning procedures over product data [CRS$^+$13, PDS05, MCT08]. The success of CAx systems pushes to expand the representation of product knowledge via formal theories, which reflect experts' conceptualisations and are accessible to computer agents. Ontologies are the state of art formal tools for these purposes [EKGT$^+$16].

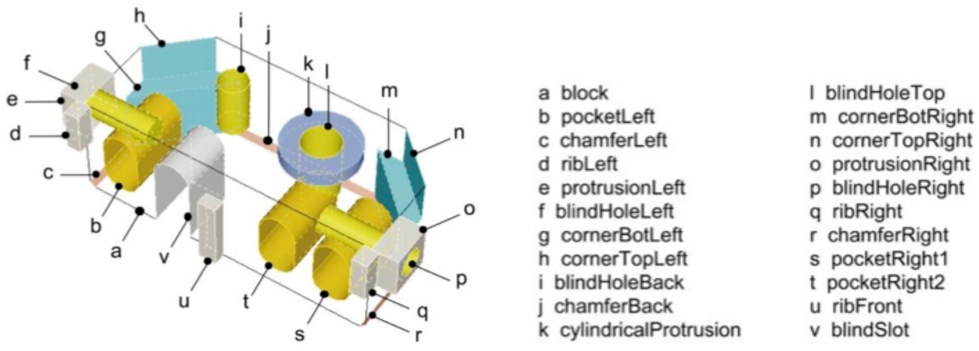
2.2.1 Feature-based modelling: An overview

Feature-based modelling is nowadays the leading approach in computer-aided product modelling [UM15].[2]

Features were introduced in the late '70s as modelling elements in CAx systems to represent and reason over both quantitative and qualitative data relevant for engineering [SM95, HT96]. The starting idea was to have (software) libraries of pre-defined elements to be reused without the need of re-defining them every time it was required. Initially, features were especially used to organise part programs for Computer Numerical Control (CNC) machines [SR88]. From this perspective, a feature like a hole or a pocket was a set of surfaces in a computer model associated to some rules for manufacturing. At the programming level, features were represented as object-oriented classes characterised by attributes; this led to develop taxonomies of features by allowing for attribute inheritance from the most general to the most specific classes within a taxonomy (e.g., [PH96]). For instance, *threaded hole feature* classified as a subclass of *hole feature* since the former, being characterised via the *threaded* attribute, is more specific than the latter. The

[2]Tang and colleagues [TCM13] talk about "feature-based informatics" to stress the impact of feature-based approaches. For conciseness, we will also talk of 'feature modelling' across the thesis (instead of 'feature-based modelling').

figure below (Fig. 2.1) shows a product model (Fig. 2.1a) with the legend of the employed features (Fig. 2.1b).



a	block	l	blindHoleTop
b	pocketLeft	m	cornerBotRight
c	chamferLeft	n	cornerTopRight
d	ribLeft	o	protrusionRight
e	protrusionLeft	p	blindHoleRight
f	blindHoleLeft	q	ribRight
g	cornerBotLeft	r	chamferRight
h	cornerTopLeft	s	pocketRight1
i	blindHoleBack	t	pocketRight2
j	chamferBack	u	ribFront
k	cylindricalProtrusion	v	blindSlot

(a) Features in a CAD model (b) Legend

Figure 2.1: Examples of feature-based product model (from [BB00])

The initial geometrical view of features was augmented with other types of information making features themselves irreducible to geometrical entities. According to ElMaraghy [ElM91] "[...] features refer to recognizable shapes which cannot be further decomposed, otherwise they will reduce to meaningless geometric entities such as lines, points, and surfaces". Di Stefano and colleagues [DSBDA04] observed that "the overall aim of feature-based representation is to convert low level geometrical information into high level description in terms of form, functional, manufacturing or assembly features." On the other hand, from the very beginning some proposed to see features as real-world constituents of products. In their seminal work, Shah and Mäntylä [SM95] employ two readings of 'feature', on the one side as an "information cluster" for the integrated representation of engineering data, on the other side as "[...] a physical constituent of a part" [ibid., p.97]. This ambiguous use of the term has not been without consequences. Brunetti [Bru03], for instance, claims that "a feature is not limited to physical elements and exists only in the world of information models" (see also [BG00]). On the other side, Nepal et al. [NSFPZ13] take features to be "meaningful real world entities to which one can associate construction-specific information". According to van Holland and Bronsvoort [VHB97], "a feature [is a] physical part of an object mappable to a generic shape and having functional significance". These remarks show that the meaning of 'feature' is not fixed, it is chosen depending on applications and contexts and, unfortunately, without a systematic treatment. Table 2.1 presents some recurrent definitions for 'feature'; its double understanding, as a modelling element and as a physical entity, emerges clearly from the definitions.

As said, historically much of the work in feature-based product mod-

Feature definitions
“A region of interest in a part model” [WP88]
“[...] modeling entities that allow commonly used shapes to be characterised [...] with a set of attributes relevant to an application” [SM95]
“An information unit describing an aggregation of properties of a product model that are relevant in the scope of a specific view on the product” [Den99]
“The characteristics of a product that result from design” [Gro07]
“[...] a physical entity that makes up some physical part” [SM95]
“[...] any geometric or non-geometric attribute of a discrete part whose presence or dimensions are relevant to the product’s or part’s function, manufacture, engineering analysis, use [...]” [DSBDA04]
“A physical constituent of a component” [Imr13].
“A physical part of an object mappable to a generic shape and having functional significance” [VHB97].

Table 2.1: Feature definitions across the literature

elling has been focused on *form features*, namely feature specifications in terms of shapes recurrently used for product development purposes like hole, slot, pocket, boss and chamfer [BBB00, PH96]. However, the exploitation of feature-based modelling to design, manufacturing or engineering analysis [UM15, HPR00, LY15, DS13], among others, led to disparate feature categories, see Table 2.2.[3] Currently, there is no shared methodology for feature classification, since it depends on application requirements and scenarios; doubts have also been raised about the possibility of an exhaustive categorisation of all feature categories [HPR00, NK07].

The dependency on applications has led to the characterisation of the very same features in different ways depending on the modelling perspective one adopts [SVHK93, WRP06, BG00]. A through hole, for instance, is classified as a *functional feature* if described from the perspective of its functionality in a product, while it is classified as a *machining feature* if emphasis is put on the operations required for the hole realisation [BJ93]. Interoperability problems arise among different classifications, because the

[3]The table is neither an exhaustive list of the categories found across the literature, nor it shows the relationships between the categories. *Assembly features*, for instance, are sometimes conceived only from a geometric perspective [Den99, MBTJ07], while in some other cases their description is enriched with manufacturing details [CMV03, VHB00, IY15].

Feature class	**Feature use**	**Example**	**Reference**
Assembly feature	Used to represent assembly knowledge	Shaft for assembly	[Den99, MBTJ07, CMV03, VHB00, IY15]
CAE feature	Used to represent engineering analysis knowledge	Stress analysis feature, fluid flow analysis feature	[Xu09, LY15, Lee05, SBOW04, HLGF10]
Form feature	Used to represent elements characterised via shape properties	Hole, pocket, chamfer	[KYDH06, SVHK93, BJ93, Han96, HPR00, CMV03, SM11, QD04]
Functional feature	Used to represent functional knowledge	Hole for assembly	[SVHK93, BJ93, WRP06, Bro03, SWS93, UIY$^+$96]
Geometric feature	Used to represent a geometric element	Surface, edge, vertice	[KYDH06, SM95, BJ93, Han96]
Machining feature	Used to represent the effects of machining processes	Amount of material swept during a drilling process	[HF06, MSS02, HHPS13, Han96, HPR00, Xu09, DS13, FOL$^+$03, ZTT12, LLCN15]
Material feature	Used to represent material properties	Ceramic feature	[SM95, QD04, SM11, UM15]
Structure feature	Used to represent components	Wall, column, screw	[BJ93, SFFK$^+$03, RRB15]

Table 2.2: A partial list of feature categories (ordered alphabetically) with use information and examples.

semantics of the represented notions changes. A hole as a functional feature, for instance, is meant as a void space in a product, sometimes called a *negative volume* [SG05]. It is because of a void that an assembly functionality can be attributed to a hole used to accommodate a screw. On the other hand, process planners reason in terms of volumes of material to be removed from the workpiece undergoing a manufacturing process. Thus, a hole as a machining feature is a part to be removed from the workpiece. Fig. 2.2 shows a feature-based product model, while the table in Fig. 2.3 shows how the features in Fig. 2.2 can be differently classified from a design or a manufacturing perspective.

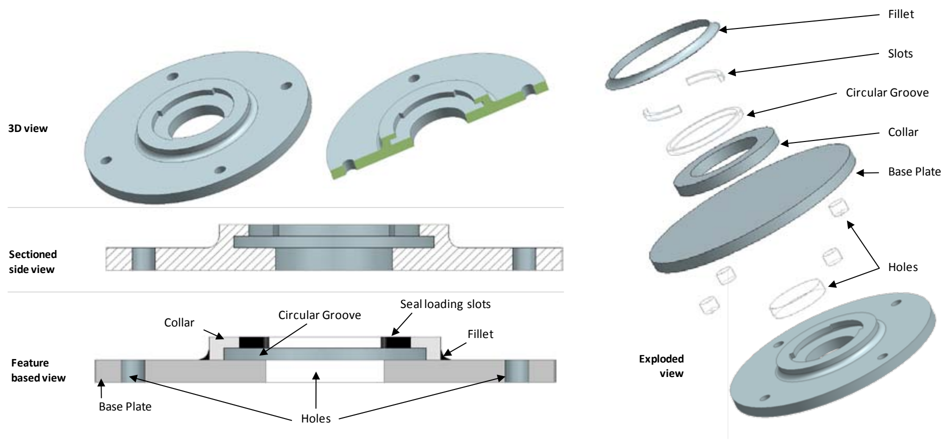
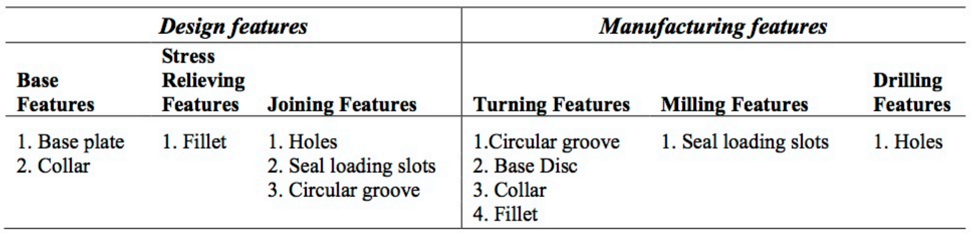


Figure 2.2: Product model (from [AHYC12a])



Design features			*Manufacturing features*		
Base Features	**Stress Relieving Features**	**Joining Features**	**Turning Features**	**Milling Features**	**Drilling Features**
1. Base plate 2. Collar	1. Fillet	1. Holes 2. Seal loading slots 3. Circular groove	1.Circular groove 2. Base Disc 3. Collar 4. Fillet	1. Seal loading slots	1. Holes

Figure 2.3: Multiple perspectives on the same features (from [AHYC12a])

Nowadays, the notion of feature is used with a variety of meanings, not only with reference to holes, bosses and the like, but “anything having an attribute of interest” [UYC$^+$13]. Features include components (sometimes called *structural features* [RRB15]), but also qualitative characteristics like colours, dimensions and shapes among others [Bro03, RGB11, SFFK$^+$03]. This generality is not surprising and from the very beginning some have proposed to understand features as any element that is relevant for product development purposes. In De Fazio and colleagues [DFEG$^+$90], for instance, a feature is “any geometric or non-geometric attribute of a discrete part whose presence [is] relevant to the products function [..]”. Recently, Pourtalebi and Horvath [PH16] propose looking at features as *complex properties* of cyber-physical systems. In this perspective, a feature is not just a product's (roughly speaking) aspect, e.g., its dimension or a hole in one of its parts; it is rather a property that is structured in (possibly simpler) further properties.

The interoperability problem raised by heterogeneous and fragmented treatments of features calls for representational methodologies that can (at least) clarify the different types of information at stake. Hopefully, we can also organise such information in a way that is coherent with the different modelling perspectives. This would allow for reliable data integration while preserving the intended semantics. Yet, at the moment this does not seem possible and each community adopts its own representational approach relying on specific application requirements, as we will see in the next sections.

2.2.2 Features in standards

Different standards have been proposed over the years to handle product data in a way that is independent from specific software formats and can support data exchange, although with several limitations [UYC$^+$13, MCT08]. We shortly present in this section two well-known standards, STEP and STEP-NC, given their wide application for feature-based product data management. We select these standards among others, because of their impact on the development of ontologies for product data management (see [Sect. 2.2.3]). We shall comment on other standardisation initiatives throughout the thesis.

The ISO standard *Automation systems and integration–Product data representation and exchange*, commonly known as STEP (ISO10303) [fSI94], is considered the most relevant effort towards the standardisation of product data [SFKS01] and the development of engineering environments for data sharing and management across the entire product life-cycle [ZHX09]. STEP provides a set of models formalised in the EXPRESS modelling language [DP94], as well as an application-independent data format to represent and share product data. According to a report of the U.S. National Institute of Standards and Technology (NIST) [GOP02], it is estimated that STEP gives the possibility of saving million of dollars by reducing interoperability problems between information systems used in industrial applications. The ISO 10303-AP203 is the main application protocol within STEP for geometric design data exchange between CADs, whereas manufacturing features are at core of the AP224.

STEP has been combined with the *Data Model for Computerized Numerical Controllers* data structure resulting in STEP-NC (ISO 14649) [fSI04] – NC standing for Numerical Control. This standard supports the integrated representation of products, machining operations and machining tools; it is thus used to link CAD and Computer-Aided Manufacturing (CAM) applications to CNC systems. The core class for feature representation in STEP-NC is *manufacturing feature*, which is understood along with STEP; see Fig. 2.4 where the attribute *its_operations* is used to define the set of machining operations required for manufacturing the feature at hand; *its_workpiece* refers to the workpiece the feature is part of; *its_id* is the unique identifier of the feature. The class *manufacturing_feature* is the superclass of *region*,[4] *two5D_manufacturing_feature* and *transition_feature*. Note that in the EXPRESS syntax, *manufacturing_feature* is modelled as an *abstract supertype* (aka superclass) in the sense that it only instantiates through its subclasses.

```
ENTITY manufacturing_feature                                  (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(region, two5D_manufacturing_feature,
  transition_feature));
  its_id:                 identifier;
  its_workpiece:          workpiece;
  its_operations:         SET [0:?] OF machining_operation;
END_ENTITY;
```

Figure 2.4: STEP-NC *manufacturing_feature* class (from [fSI04])

The application, advantages and disadvantages of both STEP and STEP-NC are well-documented across the literature. The reader can refer to [AK07, BN00, ZHX09] for reviews of STEP-based systems and approaches to feature modelling.

[4]The term ‘region’ is used in manufacturing to refer to products’ parts that are not components. We will have more to say about components in [Ch. 5].

2.2.3 Features in information models

Several initiatives focus on the development of feature specifications in terms of models in the Unified Modeling Language (UML) [RJB04], taxonomies or computational ontologies for disparate applications within the product lifecycle data modelling.

Tang and colleagues [TCM13] propose a UML meta-model, called Unified Feature Model, to manage feature-based information across CAx systems (see also [UM15]). The meta-model is built around the so-called *Generic feature* class, which aggregates quantitative and qualitative constraints, and is meant to be the most general class for representing features in engineering applications. The Unified Feature Model has been applied for different modelling scenarios [UM15, MBTJ07, YM16] and has been used as conceptual basis for further extensions [RRB15]. In the latter work, Romero and colleagues propose the so-called Feature-based Meta-Model, which shares the aim of the Unified Feature Model, but also provides a basic feature taxonomy. Both approaches, however, have not been developed by means of an ontological methodology aimed at individuating the domain entities at stake. (In [Ch. 7] we consider how the work proposed in [TCM13, RRB15] can be grounded on an ontological basis.)

The Core Product Model (CPM) [FFBS08], as the name suggests, is proposed as a *core* model for the representation of product knowledge. The CPM is specified in UML (see Fig. 2.5). In the CPM a product, called *artefact*, is represented as an aggregation of *form*, *function* and *feature* (see the figure below). The latter is related to artefact via the *hasFeature* association and is represented as the aggregation of form and function. A feature is a physical entity in a product that has a shape and a functionality.

The CPM is reused across different ontologies. Dartigues and colleagues [DGG$^+$07] propose the so-called Feature Ontology to enable data exchange between CAD and Computer-Aided Process Planning (CAPP) systems. The ontology is formalised in both UML and the Knowledge Interchange Format (KIF) [GF$^+$92]. The class *feature* is here specified as the aggregation of different constraints by which application-driven knowledge can be specified. Additionally, the Feature Ontology includes a taxonomy of form features based on STEP, covering, e.g., subtraction and protrusion features, among others. In [RHF$^+$06, FGL$^+$07, SFSW05] the CPM is specialised for assembly design applications, resulting in the so-called Open Assembly Model (OAM), which is formalised in UML. Assembly features play a fundamental role in the OAM to represent the physical means by which connection relationships can hold among the components of assembled products.

The Common Design-Feature Ontology (CDFO) [AGGSP07, AGGS$^+$14] is an ontology for CAD systems interoperability. Various feature classes (e.g., *hole*, *counterbore*, *countersunk*) are extracted from systems like Catia V5, Pro/ENGINEERING and SolidWorks, and classified into a taxonomy

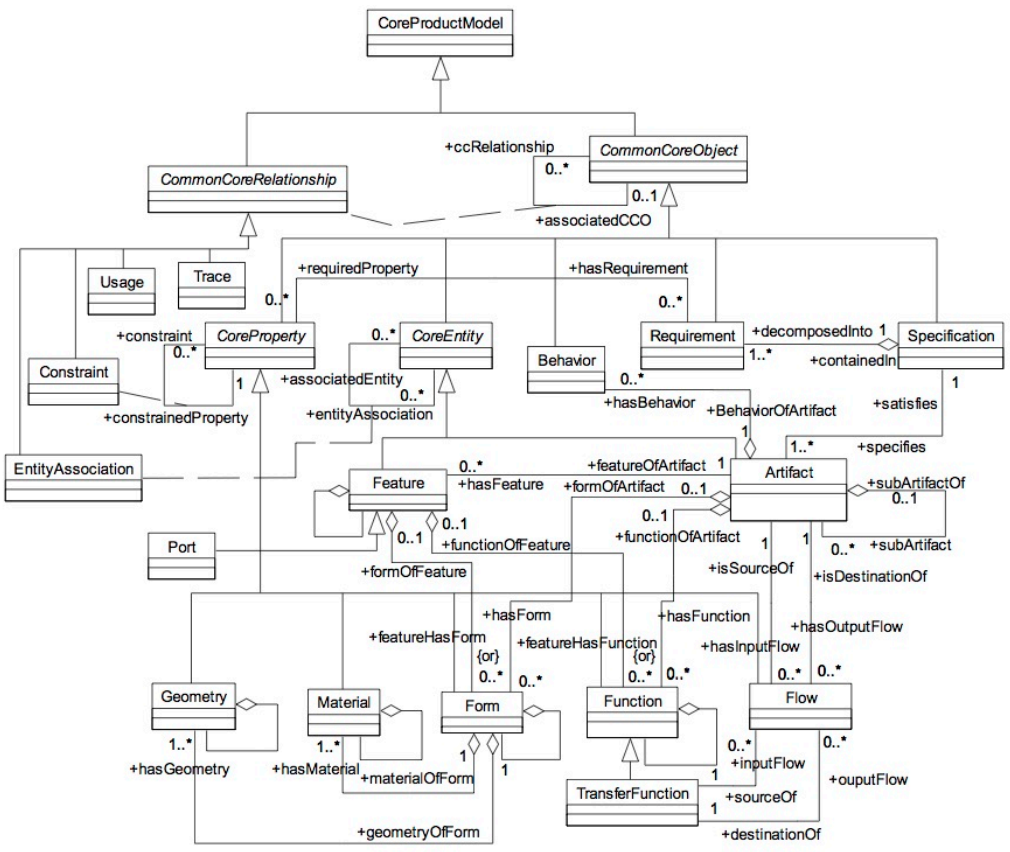


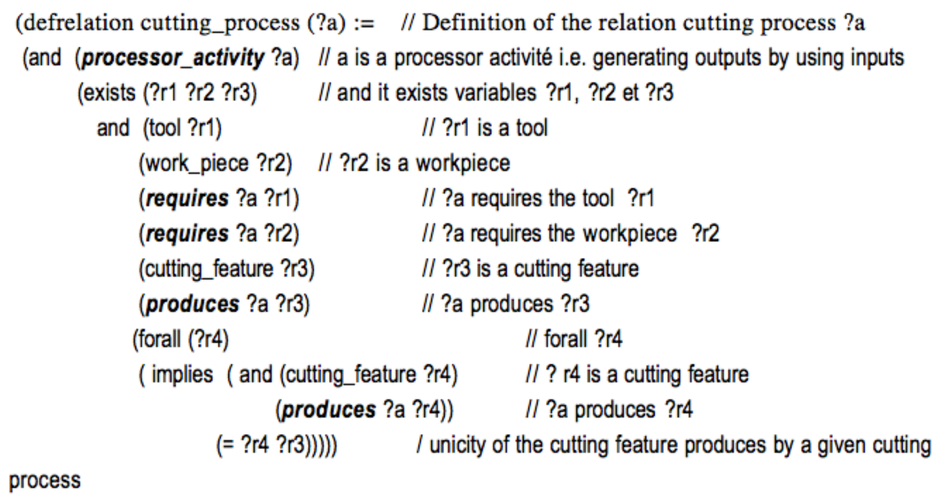
Figure 2.5: Core Product Model (CPM) (from [FFBS08])

represented in the Web Ontology Language (OWL) [MVH+04]. For example, a general class for hole features taken from CATIA is specialised via the SolidWorks classes *simple drilled*, *tapered drilled*, *counterbore drilled*, etc.

Deshayes and colleagues [DEBB05] propose a formal representation of machining processes by means of the Process Specification Language (PSL, ISO 18629) [Grü09]. The ontology includes notions like *tool*, *workpiece*, *volume* and *machine tool*, which are relevant for process representation. In this framework features result from cutting operations performed by machining tools. Most of the terms used in the ontology are only informally defined, rather than constrained in a formal manner. Fig. 2.6 shows the formal definition of the class *cutting process*, which is an event[5] that realises a so-called 'cutting feature' in a workpiece. Informally, a cutting process removes some portion of material in order to change the shape of the workpiece at hand.

Researchers at the Wolfson School of Mechanical and Manufacturing Engineering at Loughborough University propose different ontologies to deal

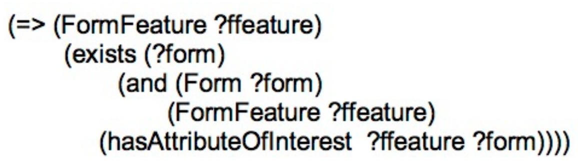
[5]We shall use 'event' as general label for process-like entities, see [Ch. 3] for details.



```
(defrelation cutting_process (?a) :=    // Definition of the relation cutting process ?a
 (and  (processor_activity ?a)  // a is a processor activité i.e. generating outputs by using inputs
        (exists (?r1 ?r2 ?r3)          // and it exists variables ?r1, ?r2 et ?r3
          and  (tool ?r1)                        // ?r1 is a tool
               (work_piece ?r2)   // ?r2 is a workpiece
               (requires ?a ?r1)                // ?a requires the tool  ?r1
               (requires ?a ?r2)                // ?a requires the workpiece  ?r2
               (cutting_feature ?r3)            // ?r3 is a cutting feature
               (produces ?a ?r3)                // ?a produces ?r3
              (forall (?r4)                                    // forall ?r4
               ( implies  ( and (cutting_feature ?r4)          // ? r4 is a cutting feature
                              (produces ?a ?r4))              // ?a produces ?r4
                     (= ?r4 ?r3)))))           / unicity of the cutting feature produces by a given cutting
process
```

Figure 2.6: KIF representation for *cutting_process* in [DEBB05]

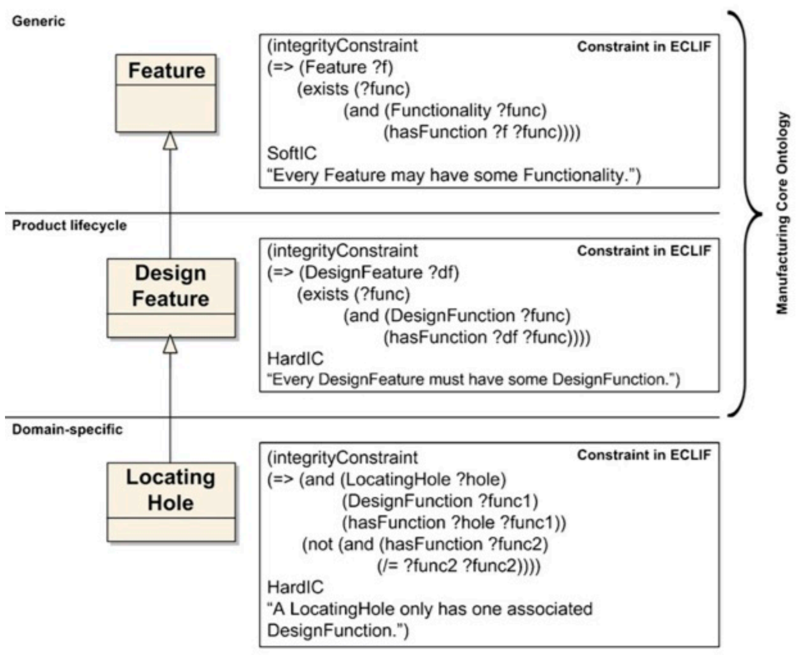
with product lifecycle information. The ontologies are formalised in the Extended Common Logic Interchange Format (ECLIF), which is the extended version of Common Logic embedded in the commercial HighFleet environment used for ontology development. As an extension of Common Logic, ECLIF allows for the formalisation of more expressive constraints than OWL, e.g., n-ary predicates. Usman and colleagues [UYC$^+$11] propose the Manufacturing Core Ontology (MCCO) as a common semantic foundation for knowledge representation in manufacturing; the ontology is extended and exploited in [AHYC12a] for manufacturability analysis and verification. Among its classes, the MCCO includes *realised part*, *part version*, *manufacturing facility* and *manufacturing process*, which are associated via different relationships. In these works the notion of feature is understood as "anything having an attribute of interest" [UYC$^+$13]; e.g., a form feature is a feature that has *form* as attribute of interest, whereas *production method* is the attribute of interest for production feature. Formally, *form feature* is represented as showed in Fig. 2.7. Although the semantics of the *hasAttributeOfInterest* relationship is unspecified, it recalls the general understanding of features as entities that are (in some sense) relevant for some application.



```
(=> (FormFeature ?ffeature)
      (exists (?form)
           (and (Form ?form)
                (FormFeature ?ffeature)
         (hasAttributeOfInterest  ?ffeature ?form))))
```

Figure 2.7: *FormFeature* class as represented in [UYC$^+$13]

In [IY15] feature-based assembly constraints are represented in KIF for data sharing in assembly design and planning. In [UY14] features are used to share product data across design and machining experts. Fig. 2.8 shows part of the UML taxonomy of feature classes proposed in [CYG+13] along with their representation in ECLIF.[6] In all these works it remains unclear how features relate to products. The authors adopt relationships like *associatedTo* [UYC+13], *holds_feature* [CY11], or *hasFeature* [Anj11] to link, e.g., a gear (product) to a hole (form feature), while the semantics of the relationships is underspecified. It is, e.g., unclear whether a hole is spatially located in a product, namely whether the spatial region where a hole locates is occupied by the product, too. Informally, the authors seem to conceive features as some sort of spatial parts. Usman, for example, claims that "the concept *Form Feature* should represent a certain portion of a component" [Usm12, p.117]. Along the same lines, a feature is (informally) defined in [Anj11] as a "distinct part of an engineering component". As we have previously seen, the understanding of features as products' parts is not uncommon across the specialised literature (see Table 2.1).



```
(integrityConstraint
(=> (Feature ?f)
    (exists (?func)
          (and (Functionality ?func)
               (hasFunction ?f ?func))))
SoftIC
"Every Feature may have some Functionality.")
```

```
(integrityConstraint
(=> (DesignFeature ?df)
    (exists (?func)
          (and (DesignFunction ?func)
               (hasFunction ?df ?func))))
HardIC
"Every DesignFeature must have some DesignFunction.")
```

```
(integrityConstraint
(=> (and (LocatingHole ?hole)
          (DesignFunction ?func1)
          (hasFunction ?hole ?func1))
     (not (and (hasFunction ?func2)
               (/= ?func2 ?func2))))
HardIC
"A LocatingHole only has one associated
DesignFunction.")
```

Figure 2.8: Example of feature representation in ECLIF (from [CYG+13])

Kim and colleagues [KYDH06] use assembly features to represent joining constraints. In this sense an assembly feature is a form feature that is functional for the assembly of components. The proposed ontology in-

[6]Axioms in ECLIF are called *soft* or *hard*. When a ECLIF ontology is populated in a knowledge base, the violation of an hard constraint stops the user, while the violation of a soft constraint just triggers a warning message.

cludes general classes like *product*, *feature* and *manufacturing [process]*, but also more specific information concerning, e.g., whether two components are welded via a butt- or T-joint. The ontology is formalised in both OWL and the Semantic Web Rule Language (SWRL) [HPSB$^+$04].

Ramos et al. [RGB11] propose an ontology-based approach for automatic feature recognition to facilitate the linkage between CAD and CAM applications. The authors provide the so-called CAD Ontology and Feature Ontology. The former specialises the general class *CAD features* in *qualitative features* and *quantitative features*. Then, *materials*, *colors* and *primitives* (e.g., *line*, *arc*) are subsumed under *qualitative features*, whereas *angle*, *point* and *parameter* under *quantitative features*. The Feature Ontology is used to model more specific classes, such as *round slot*, *circular hole*, among others. Both ontologies are expressed in OWL and are related to each other by a formal map.

In the same direction, Wang and Yu [WY14] present an ontology that is split in two modules, the STEP Box and the Feature Box. The former consists of a partial OWL formalization of the ISO10303-AP203 [138] and includes classes like *loop*, *edge*, *face* and *surface*, among others, which are the basic building blocks of feature classes. The latter ontology describes features as combinations of geometric elements as introduced in the STEP Box. For example, the feature class *through cut* is represented a set of inner wall faces that are circularly connected.

Štorga and colleagues [ŠAM10] develop the so-called Design Ontology based on the Genetic Design Model System theory [EH08, And92, AHC15]. The ontology is developed as a domain extension of the Suggested Upper Level Ontology[7] (SUMO) by using the OntoEdit tool. By means of SUMO, the ontology distinguishes between physical and abstract entities, where the former, differently from the latter, exist in space and time. Then, *form feature* is subsumed under *material object*, the latter being a subclass of *physical entity*. Despite the subsumption relationship, what the authors mean is that a form feature cannot exist without a material object.[8] Form feature is thus subsumed under material object in the sense of ontological dependence.

[7]`http://www.adampease.org/OP/`, last access on January 2017.
[8]Personal communication.

2.3 Open problems

The exploitation of feature-based approaches for the representation of features as classes of entities sharing common attributes has been focused on application concerns leaving aside the semantic clarification of feature notions. Even approaches that propose application independent frameworks treat features as aggregations of attributes and values, without addressing what a feature is meant to represent in terms of domain entities. As we previously saw, features are sometimes treated as non-physical elements, i.e., entities in product models that lack spatial properties, while in some other case they are real-world elements in physical products. The two views model important aspects that engineers need to take into account, but their integration in an information system requires careful analysis: to claim, e.g., that a non-physical feature constitutes a physical product would easily lead to logical inconsistencies. The development of computational models in languages with formal semantics (e.g., OWL) does not guarantee *per se* the clarification of the represented notions. Formal semantics is a logical tool by which the interpretations of a language can be controlled and constrained to the desired ones. However, it does not support by itself coherent and transparent knowledge representation [Gua94, BSŠT15, SBM14]. To provide a concrete example, the codification of standards in Semantic Web languages (e.g., [KBF$^+$09, BVLDV09]) surely improves the computational tractability of the developed models; differently from models in EXPRESS, for example, one can automatically reason over a STEP model in OWL. However, the mere codification does not necessarily lead to the disambiguation of the represented notions. STEP AP224, for example, treats manufacturing features as both materials to be removed from workpieces and the results of such removal [fSI06, §4.1.5]. This can lead to ambiguities, especially when software agents are in play. For example, a hole manufacturing feature can be both a negative space in a workpiece and a removed amount of material; a direct codification of the AP224 in OWL would not remove this ambiguity.

Current approaches suffer from (at least) the following drawbacks:

- Lack of a product knowledge theory to support the analysis and representation of engineering notions. To overcome this problem, distinctions between the different entities represented in information models have been introduced in an *ad hoc* manner leading to scattered and application-driven models. As we will see across the thesis, features cover qualities (e.g., shape, dimension), components, amounts of material, entities like holes, slots and ribs, among others. In order to develop robust information models, one has to identify and distinguish these entities and provide a framework by which they can be related to each other, as well as to products. This means that a formal representation of features has to be developed within a broader theory for product knowledge representation by which features can be distinguished from and related to non-feature elements.

- Hidden assumptions in the terms specification. This problem is common in specialised domains where general terms are assumed to be implicit to the domain of interest and, thus, their meaning is not explicitly stated. Examples are notions like functionality, feature, product and activity. In these cases, there is an implicit assumption that members of the community know how to understand these terms. This assumption particularly applies to standards. As noted in [UY14, UYC$^+$13], experts within the same research institution or in different departments within the same company attribute different meanings to the terms daily used for product development. It is therefore relevant that these meanings are formally captured to ensure automated data interoperability and inter-humans communication.

- Lack of an ontological characterisation of features. Feature-based models formally describe various constraints on features, especially at the morphological level, but do not model ontological constraints about what features are meant to be. Bidarra and Bronsvoort [BB00], for example, propose an approach to maintain features geometry throughout the modelling process. They do not introduce constraints to bind features to other entities: a hole can be inserted in a model without being related to some non-feature entity. Along the same lines, Wang and colleagues [WY14] axiomatise a variety of geometric constraints on features, but no rule is given to bind instances of feature classes to other entities. Ontological constraints of this kind are not only needed to explicitly characterise assumptions on what features are, but also to verify product models against experts' assumptions [IY15, AHYC12a].

- Lack of an approach that describes the various understandings of feature notions and allows for their comparison and possibly integration. This is however necessary to clarify the space of engineering features and to develop a solid comparison of different views. The approaches developed today tend to reduce features to morphological descriptions. In [UYC$^+$13], for instance, a screw hole (functional feature) in a design model is declared equivalent to a web hole (machining feature) in a manufacturing model about the same product. The rationale is that features sharing the same geometry can be identified (see [BG05] for a similar approach). Although geometric properties seem to provide an anchor to identify or at least combine features, geometry does not provide a solid base to deal with non-quantitative properties. The risk is to miss the design intents behind heterogeneous perspectives. In the example in [UYC$^+$13], instead of identifying the features, one should integrate the functional and manufacturing constraints they provide to preserve all available information.

- Lack of ontological distinctions in formal models. Approaches like [KYK08, DMK12] use mereo-topological theories (theories based on parthood and connection relationships) for the representation of assembly features. This is a promising line of research since mereotopologies are robust theories and their exploitation in design and manufacturing has been advocated for almost twenty years [GBM97, Sal02]. However, the relationship of parthood has different properties when applied to objects, processes or space; without a previous ontological distinction of the entities, the formal consequences of these theories can be easily misinterpreted.

- Erroneous use of taxonomical relationships. As we have seen, taxonomical (*is-a*) relationships are used to relate features to other classes within the same ontology, whereas the semantics of the relationship is meant to grasp, e.g., the dependency of a feature on a certain type of objects [ŠAM10]. More generally, there is not a systematic treatment of the relationships that link features to each other, or to the products that they are meant to characterise. In some case, one and the same relationship is used to relate a product to a feature, the latter understood as a product's part, and to relate a product to a shape, the latter understood as an abstract entity [Anj11].

From this list of problems emerges the need for a theory of engineering concepts, among which feature, that on the one side represents experts' perspectives and application concerns, and on the other side is clear on the domain entities at play. Otherwise said, the theory has to capture the meaning of the employed notions as used in the practise of product development and has to discriminate between different domain entities on the grounds of their ontological properties. Thus, the theory has to include an ontology for feature-based representation that, relying on fundamental distinctions like material vs immaterial object, can make sense of the heterogeneity of data in product models.

In order to fulfil this purpose, we reckon on the theoretical insights and formal methods of Ontology Engineering [SS13].

Chapter 3

DOLCE

In this chapter we introduce the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), the foundational ontology that is used across the thesis to analyse and formalise notions relevant for product modelling.

3.1 Introduction

The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [MBG$^+$03] is a foundational ontology explicitly designed to capture the ontological categories underlying natural language and common-sense thinking. DOLCE is part of our methodology for the analysis and formal representation of product knowledge, because it has been already employed for similar applications [BL07, BCGV09, FCG$^+$15, SRR16, DD15]. Additionally, its core classes are based on a worldview that meets product experts' conceptions of reality, e.g., the distinction between objects (e.g., a drilling machine), qualities (the machine's weight) and events (drilling operations).

Currently, three versions of DOLCE are available:

1. DOLCE [MBG$^+$03] comprises the entire axiomatisation of the ontology in first-order modal logic. Its taxonomy distinguishes between different categories at different levels of generality. For example, the class *perdurant* is specialised in *event*, *process*, *achievement* and *accomplishment*;

2. DOLCE-CORE [BM13] comprises only the core categories of DOLCE, i.e., *object*, *event*, *individual quality*, *region*, *concept*, *arbitrary sum* and it is formalised in classical first order logic without modality;[1]

3. DOLCE-Lite [2] consists of a partial axiomatisation of DOLCE in OWL [BCM$^+$03].

[1]Terminologically, DOLCE-CORE *object* and *event* replace DOLCE *endurant* and *perdurant*, respectively.

[2]`http://www.loa.istc.cnr.it/old/DOLCE.html`, last access October 2016.

In the following presentation we mainly refer to DOLCE-CORE, because its classes are reused throughout the thesis. It will be made explicitly when classes and formulas from DOLCE are reused. As a remark on notation, we write **DC**n for DOLCE-CORE axioms, where n is the axiom number in [BM13]; **DC**n^* is used for the axioms described but not formally given in [BM13]. We write **DL**n for axioms taken from DOLCE [MBG$^+$03] and **DL**n^* for DOLCE axioms that are slightly revised for our purposes, as we will explain. We adopt closed formulas and the variables not explicitly quantified are assumed as being universally quantified.

3.2 Particulars

DOLCE-CORE is an ontology of *particulars*, i.e., entities that exist in time, like the Pisa tower, the event of climbing Cerro Torre on a specific day performed by certain alpinists, and the weight of a reamer.

The relationship $\mathtt{PRE}(x,t)$ is used to specify the time at which a particular (PT) exists,[3] see (DC7); $\mathtt{PRE}(x,t)$ is read as "x is present at t". *Being present* – the property represented by the PRE predicate – is dissective and cumulative. Dissectivity (DC8) says that if an entity x is present at t, then x is present at all parts t' of t, whereas cumulativity (DC9) establishes that if x is present at both t' and t'', then x is present also in their (mereological) sum (Sum) t. (The predicates P and Sum refer to mereological notions, see [Sect. 3.3].) Finally, axiom (DC10) establishes that particulars are present in time.[4] Recall that *being present* distinguishes from logical existential quantification, traditionally represented with the symbol $\exists$ in classical first order logic. The former is used to refer to the time at which particulars are located, whereas the latter allows us to consider (to *quantify* over) the entities in the quantification domain but in an atemporal manner (see [Sid01, p. 59]).

DC7 $\mathtt{PRE}(x,t) \rightarrow \mathrm{TQ}(t)$

DC8 $\mathtt{PRE}(x,t) \wedge \mathtt{P}(t',t) \rightarrow \mathtt{PRE}(x,t')$

DC9 $\mathtt{PRE}(x,t') \wedge \mathtt{PRE}(x,t'') \wedge \mathtt{Sum}(t,t',t'') \rightarrow \mathtt{PRE}(x,t)$

DC10 $\mathrm{PT}(x) \rightarrow \exists t\ (\mathtt{PRE}(x,t))$

[3]'To exist' and 'to be present in time' are interchangeably used throughout the thesis.

[4]DOLCE includes abstract entities, which are neither in space nor in time; regions are examples of abstracts. DOLCE-CORE does not comprise a category for abstracts and treats regions as particulars, therefore as entities in time by (DC10). This choice is due to the idea of understanding regions, among others, as entities that can be created within certain conceptualisations and therefore exist in time. Also, note that concepts and regions, although are both represented as particulars in DOLCE-CORE, can instantiate, whereas particulars are commonly understood as entities that do not instantiate. This because particulars in DOLCE-CORE are understood in terms of presence in time, rather than instantiation.

3.3 Mereology

Mereology is the formal study of parthood relationships. It was firstly formalised in Lesniewski's *Mereology* (also called *Calculus of Manifold*) and Leonard and Goodman's *Calculus of Individuals*. Lesniewski's *Mereology* was an attempt – within foundational studies of mathematics – to deal with Russell's paradox about naive set theory. On the other hand, Goodman and Leonard embraced a nominalistic stance in philosophy and looked for a theory dispensing abstract entities and quantifying over only physical individuals. In their perspective the *Calculus* was a formal replacement of set theory as a theoretical tool for philosophical investigation. Mereology has been however developed independently from the foundational issues that led to its initial development. Nowadays, mereological theories play a relevant role in computer science for the qualitative representation of space, among other applications [CR08, CV99, Gal00].

DOLCE-CORE adopts the axioms (DC1)–(DC4) of extensional mereology.[5] By (DC1–DC3), the relationship of *parthood* is a partial order (reflexive, transitive and anti-symmetric). (DC4) is the axiom of strong supplementation, which says that if x is not part of y, there is a part z of x which does not overlap with y. The predicate *overlap* ($\mathtt{O}$) stands for the property of part sharing (DCdf1). The *mereological sum* ($\mathtt{Sum}$) of two entities, x and y, is defined as the entity s, such that anything overlaps s if and only if it overlaps either x or y. (The uniqueness of s is guaranteed by (DC4).) Finally, (DCdf1*) defines the *proper part* relationship (irreflexive, asymmetric, transitive).[6]

DC1 $\mathtt{P}(x,x)$

DC2 $\mathtt{P}(x,y) \land \mathtt{P}(y,z) \to \mathtt{P}(x,z)$

DC3 $\mathtt{P}(x,y) \land \mathtt{P}(y,x) \to x = y$

DC4 $\neg\mathtt{P}(x,y) \to \exists z\ (\mathtt{P}(z,x) \land \neg\mathtt{O}(z,y))$

DCdf1 $\mathtt{O}(x,y) \triangleq \exists z\ (\mathtt{P}(z,x) \land \mathtt{P}(z,y))$

DCdf2 $\mathtt{Sum}(s,x,y) \triangleq \forall w(\mathtt{O}(w,s) \leftrightarrow (\mathtt{O}(w,x) \lor \mathtt{O}(w,y)))$

DCdf1* $\mathtt{PP}(x,y) \triangleq \mathtt{P}(x,y) \land \neg\mathtt{P}(y,x)$

DOLCE-CORE also adopts an extensional mereology with an extra temporal parameter, which is of key relevance for representing parthood relationships between physical objects. According to (DC11), if two entities are related via *parthood* at time t, then they are both present at t, whereas (DC14) is the axiom of temporary extensionality. (DCdf3) defines the relationship of *temporary overlap*. The existence of temporary sums is not guaranteed.

[5]According to extensional mereology different entities cannot share the same parts.

[6]See [Sim87, CV99] for an overview on mereology.

DC11 $\mathtt{P}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$

DC12 $\mathtt{PRE}(x,t) \rightarrow \mathtt{P}(x,x,t)$

DC14 $\neg\mathtt{P}(x,y,t) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \rightarrow \exists z\ (\mathtt{P}(z,x,t) \wedge \neg\mathtt{O}(z,y,t))$

DCdf3 $\mathtt{O}(x,y,t) \triangleq \exists z\ (\mathtt{P}(z,x,t) \wedge \mathtt{P}(z,y,t))$

(DCdf5) defines *constant part* (CP). By (DC16) *parthood* simpliciter can be defined on the basis of *temporary parthood.*

DCdf5 $\mathrm{CP}(x,y) \triangleq \exists t\ (\mathtt{PRE}(x,t)) \wedge \forall t(\mathtt{PRE}(x,t) \rightarrow \mathtt{P}(x,y,t))$

DC16 $\exists t\ (\mathtt{PRE}(x,t)) \rightarrow (\mathrm{CP}(x,y) \leftrightarrow \mathtt{P}(x,y))$

3.4 Qualities

Everyday objects can be compared in a number of ways on the basis of their "characteristics".[7] For example, a drill weighs 1,5 kg, whereas a table weighs 10kg, therefore the drill is less heavy than the table; Cerro Torre is 3128mt high, whereas K2 is 8609mt, therefore the former is shorter than the latter; John's and Mary's eyes are blue, therefore they are both blue-eyed. The representation of what we roughly called "characteristics" is based in DOLCE-CORE on the distinction between *individual quality*, *quality kind* and *quality space.*

Individual quality. An individual quality is the quality of a specific particular and characterises only that particular. In this sense, even if John's and Mary's eyes are both blue, the colour of John's eyes differs from the colour of Mary's eyes, because they are the two individual qualities of two different persons.

In DOLCE-CORE the relationship between an individual quality and the entity it characterises is called *inherence* (D20); $\mathtt{I}(x,y)$ is read as "x inheres in y", where x is an individual quality and y is called the quality's *bearer.* An individual quality is ontologically bound to its specific bearer in the sense that (*i*) it cannot migrate across different particulars (DC21) and (*ii*) it cannot exist without the existence of the bearer (DC22). Axiom (DC23) states that qualities exist during the whole life of their bearers. Individual qualities are similar to *tropes* [Lou98], the difference being that individual qualities in DOLCE-CORE can change across time, whereas tropes are substituted [BM13, p.368]. For instance, a particular object *ob#* may be red at time t and green at t'. In this case, one would commonly assume in a trope theory that two different tropes, tr_{rd} (the red trope) and tr_{gr} (the green trope), inhere in *ob#* but at different times; when *ob#* undergoes a

[7]We rely hereby only on an intuitive interpretation of objects, whereas their ontological understanding within DOLCE-CORE is presented in [Sect.3.6].

colour change from red to green, tr_{gr} substitutes tr_{rd}. On the other hand, quality change is understood in DOLCE-CORE in terms of location of the same individual quality in different regions within the same quality space. This idea will be made clear throughout the section.

DC20 $\mathtt{I}(x, y) \to \mathrm{Q}(x)$

DC21 $\mathtt{I}(x, y) \wedge \mathtt{I}(x, y') \to y = y'$

DC22 $\mathrm{Q}(x) \to \exists y\ (\mathtt{I}(x, y))$

DC23 $\mathtt{I}(x, y) \to \forall t(\mathtt{PRE}(x, t) \leftrightarrow \mathtt{PRE}(y, t))$

Quality kind. Individual qualities are grouped into n non-empty quality kinds, e.g., the colour quality kind or the weight quality kind. Thus, given a quality x, we write an index i, with $1 \leq i \leq n$, such that $\mathrm{Q}_i(x)$ is read as "the individual quality x is of quality kind i", see (DCdf2*). The grouping of qualities into quality kinds may depend on metaphysical or cognitive assumptions. In the first sense, one can assume that qualities – as they exist in reality – partition into kinds. In the second sense, which we assume in accordance to DOLCE cognitive bias, the grouping of qualities into kinds may rely on different considerations, e.g., human's sensory system and cognition, or measurements methods.

Axiom (DC24) states that an entity can have at most one individual quality for each specific quality kind. In this perspective it remains to be clarified what it means for an object to be, e.g., multi-coloured (see Example 3 below).

DCdf2* $\mathrm{Q}(x) \triangleq \bigvee_i \mathrm{Q}_i(x)$

DC24 $\mathtt{I}(x, y) \wedge \mathtt{I}(x', y) \wedge Q_i(x) \wedge Q_i(x') \to x = x'$

Quality space. Qualities of the same kind can be organised into taxonomies or more sophisticated arrangements, from orderings (e.g., for weights and lengths) to complex topological or geometrical structures (the colour spindle or the taste tetrahedron, see [Gär04]). These structures are called *quality spaces* in DOLCE-CORE and consist in a formal variation of *conceptual spaces* as proposed by Gärdenfors [Gär04]. The idea is that qualities can be differently conceived and represented according to disparate principles such as measurement tools or cultural systems [MB05]. Temperature, for instance, can be measured via the Celsius or the Fahrenheit scale, as well as lengths via the International Metric System (IMS) or the British Imperial System (BIS). Therefore, each conceptualisation provides its own principles for organising qualities, e.g., grams in the IMS and grains in the BIS for mass qualities.[8]

[8]An approach similar to quality spaces is adopted in [Chu10, p.79] to provide the semantics of measurement values for ontology-based modelling in manufacturing.

In DOLCE-CORE each quality kind Q_i can be associated to one or more spaces; we write SP_{ij} for quality spaces, where the first index i refers to a quality kind and the second index j to the corresponding space. For example, SP_{ij} and SP_{ik} stand for the spaces SP_j and SP_k associated to the quality kind Q_i. Entities in a space are called *regions*; since spaces are disjoint, regions are partitioned into disjoint spaces.

Axiom (DC28) is used to establish that regions do not change over the time when they exist. The *location* relationship (L) provides the association between qualities and quality spaces; $\mathtt{L}(x, y, t)$ is read as "the individual quality y is located in the region x at time t" (DC30). By (DC31), the location of a quality at time t implies the presence of y at t. Each individual quality has to be located in (at least) one of the spaces SP_{ij} associated to the corresponding quality kind (DC34–DC35). By (DC36), the location of a quality in a single space is unique.

DC28 $\mathrm{R}(x) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(x,t') \rightarrow \forall y(\mathtt{P}(y,x,t) \leftrightarrow \mathtt{P}(y,x,t'))$

DC30 $\mathtt{L}(x,y,t) \rightarrow \mathrm{R}(x) \wedge \mathrm{Q}(y)$

DC31 $\mathtt{L}(x,y,t) \rightarrow \mathtt{PRE}(y,t)$

DC34 $\mathtt{L}(x,y,t) \wedge Q_i(y) \rightarrow \bigvee_j \mathrm{SP}_{ij}(x)$

DC35 $\mathrm{Q}(y) \wedge \mathtt{PRE}(y,t) \rightarrow \exists x\ (\mathtt{L}(x,y,t))$

DC36 $\mathtt{L}(x,y,t) \wedge \mathtt{L}(x',y,t) \wedge \mathrm{SP}_{ij}(x) \wedge \mathrm{SP}_{ij}(x') \rightarrow x = x'$

Example 1. Consider two drilling machines, dm_1 and dm_2, each one being 2,5kg heavy. In DOLCE-CORE for an object x to have a certain weight means that there is an individual quality y inhering in x, such that y is of the quality kind weight (Q_{wg}) and y is located in a region (2,5kg in our example) of the weight-space (SP_{wg}).[9] In this sense, the quality space provides a structure to attribute values to individual qualities. See (f1), where $Weight(x, 2.5kg, t)$ is read as "x weighs 2,5kg at t".[10]

f1 $Weight(x, 2.5kg, t) \triangleq$
$$\exists y\ (\mathtt{I}(y,x) \wedge \mathrm{Q}_{wg}(y) \wedge \mathtt{L}(2.5kg,y,t) \wedge \mathrm{SP}_{wg}(2.5kg))$$
(example: x weighs 2.5kg at t means that there is a quality y inhering in x, such that y is of the weight quality kind and y is located in the 2.5kg region within the weight quality space)

Accordingly, when two drilling machines dm_1 and dm_2 have the "same" weight, this means that their *different* individual qualities are located in the same region within the weight-space.

[9]For the sake of the example we assume that each quality kind is associated to only one quality space.

[10]We shall use **f***n* throughout the thesis to write axioms and definitions for examples.

Example 2. Consider a particular object that undergoes a colour change, e.g., it is red at time t and green at t'. Recall that in DOLCE-CORE for an entity x to undergo a quality change means that there is an individual quality v of kind Q_i inhering in x such that v is located in two different regions, y and z, of the space SP_j associated to Q_i. See formula (f2), where $QualityChange(x, y, z)$ is read as "x undergoes a quality change from y to z".

f2 $QualityChange(x, y, z) \triangleq$
$$\exists vtt' \bigvee_{ij}(\mathtt{I}(v, x) \wedge \mathrm{Q}_i(v) \wedge \mathtt{L}(y, v, t) \wedge \mathtt{L}(z, v, t') \wedge \neg(y = z) \wedge \mathrm{SP}_{ij}(y) \wedge \mathrm{SP}_{ij}(z))$$
(example: x undergoes a quality change from y to z means that there is an individual quality v of kind i that inheres in x, such that v is located in the region y at t and in the region z at t' within the quality space j associated to i)

For example, the formula $QualityChange(ob_1, red, green)$ represents the particular object ob_1 undergoing a colour change from red to green; by (f2), this means that the individual colour of ob_1 is located in the *red* and *green* regions but at different times.

Example 3. Consider a multi-coloured object, namely, an object that is half red and half green. Recall by axiom (DC24) that an entity can have at most one individual quality per quality kind. The example can be represented in (at least) two different ways.

(i) We may ascribe the different colours to different parts of the object x being considered, that is, x is red in y and green in z, where y and z are proper parts of x, see (f3). In this approach, we need to assume that x reduces to the sum of y and z, otherwise there might be parts of x that are neither red nor green.

f3 $RedGreenColoured(x, t) \triangleq \exists yzvw\ (\mathtt{Sum}(x, y, z) \wedge \mathtt{I}(v, y) \wedge \mathtt{I}(w, z) \wedge \mathrm{Q}_{cl}(v) \wedge \mathrm{Q}_{cl}(w) \wedge \mathtt{L}(red, v, t) \wedge \mathtt{L}(green, w, t) \wedge \mathrm{SP}_{cl}(red) \wedge \mathrm{SP}_{cl}(green))$
(example: x is both red and green at t means that x is the mereological sum of two parts, y and z, such that y is red and z is green)

(*ii*) We can refer to a quality y of x such that y is located in the region which is the sum of the red and green regions within the colour quality space.

f4 $\mathit{RedGreenColoured}(x,t) \triangleq \exists ys\ (\mathtt{I}(y,x) \wedge \mathrm{Q}_{cl}(y) \wedge \mathtt{L}(s,y,t) \wedge \mathtt{Sum}(s,red,green) \wedge \mathrm{SP}_{cl}(s))$
(example: x is red and green coloured at t means that the quality y inhering in x is located in the mereological sum of the red and green regions within the colour quality space)

First, note that colours are summative, in the sense that the mereological sum of two colours is still a colour. Second, the sum of two colours is not a new colour but a region that comprises both of them.

3.5 Concepts and roles

DOLCE-CORE includes concepts (CN) amongst particulars (DC1*). Formally, concepts are reified in the domain of quantification, therefore treated as individuals. The *classification* relationship is used to talk about the instances of a concept; see (DC17) where $\mathtt{CF}(x,y,t)$ is read as "concept x classifies y, which is present at time t", or (equivalently) "y, as it exists at t, satisfies a concept x". Classification thus implies the presence at t of the classified entity (D18). (Note that the temporal parameter in CF does not refer to the time at which the classification is done but to the time at which y is present.)

DC1* $\mathrm{CN}(x) \rightarrow \mathrm{PT}(x)$

DC17 $\mathtt{CF}(x,y,t) \rightarrow \mathrm{CN}(x)$

DC18 $\mathtt{CF}(x,y,t) \rightarrow \mathtt{PRE}(y,t)$

Concepts can be adopted to represent properties that have a contextual or dynamic nature. In [MVB$^+$04] they are employed to represent *roles.* These are properties created within social contexts in accordance to some conventions; roles are thus dynamic, anti-rigid and relationally dependent (see also [VBM08]). Dynamic means that they are played by particulars at certain times; e.g., John has the role of *being a student* at t, whereas he has the role of *being a professor* at t'. Anti-rigidity means that it is not necessary for a particular to play a role, so that acquiring or loosing a role does not affect its identity. Finally, relational dependence is a form of dependence holding between properties; in [MVB$^+$04] it is understood in terms of *definitional dependence*, that is, a property ϕ is definitionally dependent on a property ψ, if any definition of ϕ involves ψ. Roles can thus be defined on the basis of relationships whose arguments are characterised by certain desired properties, namely the properties they are meant to satisfy according to some background conceptualisation. This can be represented by defining the predicate that corresponds to a role. For example, students are (roughly said) persons enrolled at schools. Thus the role of *being a student* (*Student* for conciseness) can be defined in terms of *Person*, *School* and the relationship *enrolled*, see (f5).

f5 $Student(x,t) \triangleq Person(x) \wedge \exists y\ (School(y) \wedge enrolled(x,y,t))$
(example: x is a student at t means that x is a person who is enrolled at a school at t)

If roles are treated as concepts, *Student* has to be reified in the domain. For example, we can say that x instantiates the concept *student*[11] if and only if it is a person who is enrolled at a school (f6).[12]

f6 $\mathtt{CF}(student,x,t) \leftrightarrow Person(x) \wedge \exists y\ (School(y) \wedge enrolled(x,y,t))$
(example: x is classified by the concept *student* if and only if x is a person enrolled at a school)

3.6 Objects and events

Along with a common sense conceptualisation of reality, DOLCE-CORE distinguishes between objects and events. In philosophy, an object is sometimes understood as an *endurant* (also said *continuant*), i.e., an entity that (*i*) undergoes some change through time while preserving its own identity, (*ii*) is extended in space and (*iii*) is wholly present at every time t in which it exists, namely all its spatial parts at t are present. On the other hand, events are called *perdurants* (also *occurrents*); these are particulars that unfold in time and at any time t in which they exist, only their temporal parts at t are present (see [Sim87] on the endurant/perdurant distinction).

Philosophers lively debate whether the (fundamental) inhabitants of reality are either endurants or perdurants. Four-dimensionalism (4D), for instance, assumes a basic ontology of perdurants [Sid01], where objects are "spatio-temporal worms" that happen in time by loosing and accumulating temporal parts. Take John, a particular object. In a 4D perspective, at each instant of time t at which John exists, he exists only in his temporal part that is present at t.[13] The whole John, who existed from t (his birth) to t' (his death), is the sum of all the temporal parts that existed during the t-t' interval. Three-dimensionalism (3D) argues that objects are endurants with

[11]We employ lowercase labels for reified concepts.

[12]Note that (f6) defines the concept of *student* in an extensional manner, namely by referring to the classified individuals in the domain of quantification. However, concepts may be defined only intensionally, therefore without relying on the entities they classify.

[13]A temporal part x of y is a part of y that exists only at a certain time t and overlaps every spatial part of y that exists at t, namely it is a maximal part of y with respect to time [Sid01].

spatial but not temporal parts. In this perspective events are not always admitted in the inventory of reality. According to [Low06], for example, “an event occurs when a particular object takes on a particular property [...]”. An example is “a particular object [that] changes from being round to being oblate in shape”. In this view, events do not exist as particulars; they rather reduce to objects satisfying different properties at different times.

DOLCE-CORE avoids reducing one category to the other one on the ground of common sense. Moreover, the distinction between objects and events is assumed in different disciplines, from medicine to design, manufacturing and geoscience. Thus, the lack of recognising both categories of entities may threaten the applicability of the ontology itself.

Axiom (DC2*) represents objects (OB) as particulars with a spatial quality (SQ). Along the same lines, (DC46) captures the notion of event (E) as the bearer of a temporal quality (TQ), which locates the event in time (see DC45).

DC2* $\mathrm{OB}(x) \rightarrow \exists y\ (\mathrm{SQ}(y) \wedge \mathtt{I}(y,x))$

DC46 $\mathrm{E}(x) \rightarrow \exists y\,(\mathrm{TQ}(x) \wedge \mathtt{I}(y,x))$

DC45 $\mathrm{TQ}(y) \wedge \mathtt{L}(x,y,t) \rightarrow x = t$

Objects and events are bounded to each other via the *participation* relationship; see (DC39, DC40, DC41) where $\mathtt{PC}(x,y,t)$ is read as “object x participates in event y at time t”. Axiom (DC42) states that if an object x participates in an event y, which is part of a larger event y', then x participates in y'; according to (DC43), all parts x' of x participate in y. Finally, if a quality kind is related to events, it cannot be related to objects, and vice versa (DC44).

DC39 $\mathtt{PC}(x,y,t) \rightarrow \mathrm{OB}(x) \wedge \mathrm{E}(y)$

DC40 $\mathrm{OB}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y\ (\mathrm{E}(y) \wedge \mathtt{PC}(x,y,t))$

DC41 $\mathrm{E}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y\ (\mathrm{OB}(y) \wedge \mathtt{PC}(y,x,t))$

DC42 $\mathtt{PC}(x,y,t) \wedge \mathtt{P}(y,y',t) \wedge \mathrm{E}(y') \rightarrow \mathtt{PC}(x,y',t)$

DC43 $\mathtt{PC}(x,y,t) \wedge \mathtt{P}(x',x,t) \rightarrow \mathtt{PC}(x',y,t)$

DC44 $\mathtt{I}(x,y) \wedge \mathrm{E}(y) \wedge Q_i(x) \wedge \mathtt{I}(z,v) \wedge \mathrm{OB}(v) \wedge Q_j(z) \rightarrow \neg Q_j(x) \wedge \neg Q_i(z)$

We shall now consider the extension of DOLCE-CORE with some high-level classes that are useful for our purposes.

3.7 Material and immaterial objects

In the full DOLCE ontology a material object is an object that is *constituted* by some *amount of matter*, where the distinction between material objects and amounts of matter is based on both unity and persistence conditions. A material object is a *whole* entity under some form of (e.g., topological or functional) connection, whereas (the parts of) amounts of matter (e.g., sand or gold) are not connected into wholes, at least they are not connected in the topological sense. Additionally, DOLCE assumes that a material object can acquire or loose some of its parts while remaining the same object; on the contrary amounts of matter undergo a change in their identity when their parts change.

The *constitution* relationship ($\mathtt{K}$) holds between amount of matter (M) and object (DL1*);[14] $\mathtt{K}(x, y, t)$ is read as "x constitutes y at time t". By (DL24) and (DL25) *constitution* is asymmetric and transitive, respectively; axiom (DL26) relates *constitution* to the property of *being present.*

DL1* $\mathtt{K}(x, y, t) \rightarrow \mathrm{M}(x) \wedge \mathrm{OB}(y) \wedge \mathrm{TQ}(t)$

DL24 $\mathtt{K}(x, y, t) \rightarrow \neg\mathtt{K}(y, x, t)$

DL25 $\mathtt{K}(x, y, t) \wedge \mathtt{K}(y, z, t) \rightarrow \mathtt{K}(x, z, t)$

DL26 $\mathtt{K}(x, y, t) \rightarrow \mathtt{PRE}(x, t) \wedge \mathtt{PRE}(y, t)$

We define a material object as a physical object that is *generically* constituted by some amount of matter (DL2*).[15] *Generic constitution* means that at every time at which a material object exists, it is constituted by some matter, even though the latter may change over time.[16]

DL2* $\textsc{MaterialObject}(x) \triangleq \forall t\ (\mathtt{PRE}(x, t) \rightarrow \exists y\ (\mathtt{K}(y, x, t))$

By (Def1), we introduce immaterial objects as objects that are not constituted by any amount of matter. (This definition is not given in DOLCE/DOLCE-CORE, therefore it is hereby introduced with a different enumeration.) As we will see in [Ch. 6], this notion is useful to talk about hole-like entities.

By axiom (Ax1), we establish that *object* covers only immaterial objects, material objects and amounts of matter. By (Ax2), *amount of matter* is disjoint with both *immaterial object* and *material object.* Finally, by (Ax3) *material object* and *immaterial object* are disjoint.

[14]Axiom (DL1*) is based on DOLCE (Ad20) in [MBG+03].

[15]Axiom (DL2*) is a formal variation of DOLCE (Ad30) [MBG+03]. We write 'material object' instead of using an acronym, because it is a notion hereby introduced that does not belong to DOLCE-CORE. The same for 'immaterial object' in (Def1).

[16]DOLCE distinguishes between *generic* and *specific* constitution; differently from the former, in the latter case an object is constituted always by the same material; refer to [MBG+03] for details.

Def1 $\textsc{ImmaterialObject}(x) \triangleq \mathrm{OB}(x) \wedge \forall t\ (\mathtt{PRE}(x,t) \rightarrow \neg\exists y\ (\mathtt{K}(y,x,t))$

Ax1 $\mathrm{OB}(x) \leftrightarrow \textsc{ImmaterialObject}(x) \vee \textsc{MaterialObject}(x) \vee \mathrm{M}(x)$

Ax2 $\mathrm{M}(x) \rightarrow \neg(\textsc{ImmaterialObject}(x) \vee \textsc{MaterialObject}(x))$

Ax3 $\textsc{ImmaterialObject}(x) \rightarrow \neg\textsc{MaterialObject}(x)$

DOLCE includes the *feature* predicate, which refers to objects that cannot exist in space and time if detached from some other object. Examples are (material) surfaces, stains and bumps. In philosophy features are known as *parasitic entities* [CV94], a wording that emphatically stresses their dependence on the objects to which they are related, known as features' *hosts.* Features are of key relevance for our work; therefore we will dig into their conceptual analysis and formal representation in [Ch. 5] and [Ch. 6].

The figure below (Fig. 3.1) shows the taxonomy of the classes presented throughout the chapter (the arrows between the classes are subsumption relationships).[17]

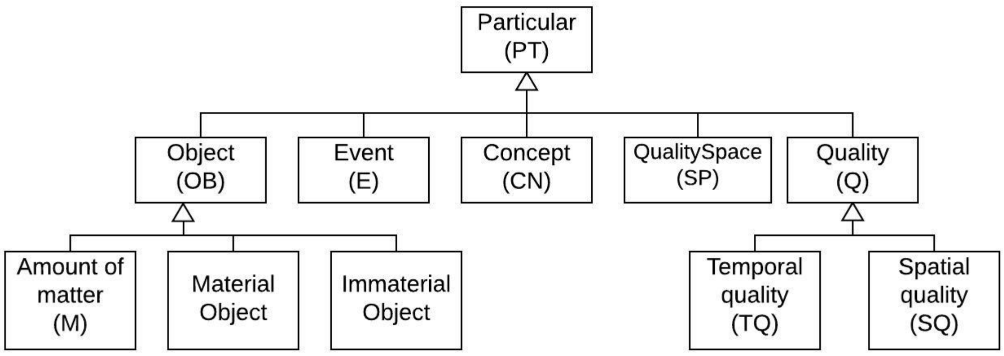


Figure 3.1: Upper-level classes of DOLCE-CORE adapted for our purposes

[17]The DOLCE-CORE taxonomy shown in the figure does not comprise the class *arbitrary sum*, which is not relevant for our study.

Chapter 4

Technical products and product types

In this chapter we present the notions of technical artefact and product type by exploring also the relationship of *compliance* between them. By the end of the chapter we compare our approach with some relevant studies. We motivate our approach by relying on both the engineering and the philosophical literature, as well as on formal ontologies for product knowledge representation.

4.1 Artefacts

Most of the objects we daily interact with are the result of human craft. The houses we live in, the buses and airplanes we catch to reach our destinations, laptops, chairs, zebra crossings, bridges, bikes are just a few examples. At a larger scale, airports, nuclear factories, dams, universities, space-stations and entire metropolis are purposely invented by humans. Interestingly, anthropologists and historians "measure" the evolution of human civilisations also on the basis of our ancestors' abilities to handle materials and produce the items they needed to survive (e.g., the Stone and Bronze ages). There is no surprise that the study of *artefacts* – the general category employed to cover created objects – is at the focus of different disciplines, from history, anthropology and ethology, to philosophy, cognitive science and applied ontology (see [Hil11, Kro12, VCBG13] for further references).

Since (at least) Aristotle, artefacts are contrasted to *natural objects*, particulars which are not created by human intervention. The distinction is however rough and it is challenging to individuate the red line between artefacts and natural objects. Is a genetically modified plant a natural or artefactual object? Is a pebble used as paperweight an artefact? Is a bird nest an artefact? These are only some of the questions artefact theories need to deal with [Kro12, HV09].[1]

Independently from the notion of natural object, the category of (human-made) artefact is broad enough in itself. Some artefacts, for example, require some sort of social convention, because they exist as far as there is some agreement among the people living in a community [Sea95]. This consideration, admittedly general, applies, e.g., to laws and salaries but also to airports and factories, among others, which are socially regulated. A further challenge in the analysis of the latter artefacts, called *socio-technical systems* (STS) [BS11], is that they involve *organisations* in which humans interact with each other, as well as with machines. STSs thus require an analysis of social roles, duties and responsibilities that regulate such interactions. Artefact theories cover also works of art, understood as artefacts made for aesthetic rather than practical purposes [Dip93]. Modern art, however, reminds us the slippery distinction between art and non-art; Duchamp's *Fountain* is a notorious example.

For the purposes of our work, we leave behind most of the issues that artefact theories need to face and focus on engineering artefacts, commonly called *products*. Also, because our research strives from the need of providing the foundations for ontologies about designed and manufactured material objects, we leave aside services and software from our analysis.[2]

4.1.1 Products

The Industry Foundation Classes (IFC) provides the following comment on the notion of product:[3] "Products include manufactured, supplied or created objects [...]. Products are defined by their properties and representations". This view is widely shared across engineering.

First, note the tight relationship – usually called *compliance* [Gal98] – between products and the properties they are required to satisfy. According to IFC, indeed, a product is "defined by" its properties, which are intentionally identified in designing activities to meet customers' requirements. We call these properties *design properties* throughout the thesis. Second, although IFC informally distinguishes between "manufactured", "supplied" and "created" products, it remains unclear how these distinctions apply. In particular, it is unclear whether supplied products are meant to undergo some production process. However, given the explicit (albeit ambiguous) reference to "manufactured" and "created" products in the standard, it is

[1]Interestingly, the U.S. Food and Drug Administration (FDA) recently faced the definition of 'natural' for food labelling by recalling some of the issues addressed in artefact theories about the natural/artefactual distinction; see the FDA website, entry: *"Natural" on Food Labeling*, last access October 2016.

[2]See [FG08, WGGM14, Kas10] for readings on these topics.

[3]See `IfcProduct` at `http://www.buildingsmart-tech.org/ifc/IFC4/Add1/html/`, last access October 2016.

plausible to consider "supplied" products as not being human-made. For instance, coal is mined but its existence does not rely on design efforts. There is, indeed, a sense in engineering in which natural objects are products, too. For instance, according to the STEP standard (Part 1) [fSI94] a product is "[a] thing or substance produced by a natural or artificial process". The standard does not explain its conceptualisation; it nevertheless recognises natural objects within the range of products.

In some other cases what is a product is not fixed but it depends on application contexts. For instance, a company that produces automobiles may buy engines, wheels and brakes from different producers and assemble them together; thus the company may consider the automobiles as products, whereas the engines, wheels and brakes as resources. In this perspective, a product is something that a company produces and sells, rather than buys (see, e.g., [PUPS$^+$16]). Therefore, one and the same particular is a product for the producing or selling company and a resource for the buying company.

4.1.2 Compliance

From a general perspective, we characterise compliance as the relationship that holds between an object x and a list of design properties, call it ϕ, if and only if x satisfies ϕ. This list of properties is differently called across the literature; we call it *product type.*

In [Sect. 4.2] the notion of product type is presented in more details. For the sake of clarity, we note that a type is a complex entity provided with a structure, that is, it is not just a plain list of properties but different constraints and relationships hold between the properties in a type. For example, a product type may establish that if the dimension of the designed product varies within the pre-defined tolerances, the dimensions of the product components have to be accordingly adjusted.[4]

It is also reasonable to assume that a product type satisfies a unity criterion by which its properties form a *whole.* We shall not enter into the specification of these unity criteria, since it would require an analysis of the (meta-) properties that properties lists have to satisfy to be product types. We shall now turn to compliance.

Artefact theories for engineering commonly ask whether non-compliant objects are products at all [BFG$^+$14]. On the one side, one may answer negatively to this question if the object at hand does not satisfy the design properties it was meant to satisfy. On the other side, this view is too strict. For example, it is nowadays common to find shops specialised in selling items, especially clothes, which have production defects. Perhaps, one would not pay thousands of Euros for a Valentino suit which is badly sewed and

[4]It goes behind the purposes of this work to enter into the details of the structures of types. We introduce in [Ch. 6] the formal axioms that allow for the modelling of types from a general perspective.

likely the Valentino company would not even put its logo on such a suit. Nevertheless, the suit may be dressed and it would be odd to not consider it as a product. Clearly, the assumption is that the product still satisfies some of its design properties, if it can be dressed although the defects.

From this perspective, one may distinguish between (at least) two different senses of compliance, i.e., *total* and *partial* compliance. In the first case, an object x is totally compliant with ϕ, if and only if x satisfies all the design properties in ϕ (within the pre-defined tolerances). In the latter case, an object x is partially compliant with ϕ means that x is compliant only with some of the design properties in ϕ. For example, Valentino suit was meant to be grey and 16 sized, whereas it is grey and 12 sized.

The question is now whether totally non-compliant objects and partially compliant objects are products.

In the second case, the answer likely relies on the design properties being satisfied. For instance, Valentino stylists may design men suits of the Spring Collection 2016 as being grey, made of cotton and produced in a specific tailor factory. Therefore, if the production fails in meeting one of these properties, the designers have to make a choice about whether the created items are Valentino suits of that collection. The example suggests that there are some design properties, upon which experts agree, that an object has to necessarily satisfy to be the realisation of a certain product type. We call them *essential* (or *mandatory*) design properties along with [EH08, JGAB09]. On their grounds we define the notion of *strict compliance* as the relationship that holds between an object x and a product type ϕ, when x satisfies the essential design properties in ϕ.[5]

In the case of non-compliance, an object does not satisfy (by definition) any property of the corresponding product type. It is therefore unlikely that it can be considered as the physical realisation of the type. It is however hard to deny, if possible at all, that it exists because of intentional human creation.

[5]It is up to experts to decide which (design) properties in a product type are essential, therefore strict compliance can be either total or partial, depending on whether all or only some of the properties are essential. Also, along with [Gua14, HV09, Tho14] we do not assume functionalities as being necessarily essential properties.

4.1.3 Taxonomy of artefactual notions

From the considerations above we distinguish between:

1. *Technical product*, a material object that complies (at some degree) with a corresponding product type and is the outcome of intentional production events;[6]

2. *Technical role*, the role of a material object which is attributed with a property that is relevant for someone's purpose and is accordingly used.

Therefore, *being a technical product* is a production- and design- dependent property. 'Production-dependent' means that an object is a technical product if it is *intentionally* created.[7] 'Design-dependent' is to say that *being a technical product* means to comply with a product type. As a consequence, a produced object that does not satisfy the design properties deemed to be essential by the corresponding designers is not a technical product. However, it is still an *artificial object*, which distinguishes from a natural object because it is human-made, although it fails in satisfying the design properties it was meant to comply with. We call it *defective artefact*. Finally, *by-products* are physical entities that result from intentional production events, even though the events are not performed for their creation, e.g., the material waste that results from cutting a plank. In artefact theories, it is commonly said that by-products are not intentionally produced, because they are not at the intentional focus of the agent who performs the production event leading to their creation [BFG$^+$14].

Fig. 4.1 represents the taxonomy of the notions discussed throughout this section. We use the general notion of *artificial entity* to refer to created particulars in opposition to natural particulars. *Artificial entity* specialises in both *artefact* and *by-product*. Differently from a by-product, an artefact is (generically) understood as an intentionally created entity. *Artefact* covers both *technical product* and *defective artefact*. As said, technical products are both intentionally created and compliant (at some degree) with their product types, whereas the latter constraint does not hold for defective artefacts. (The taxonomy is not meant to be exhaustive.)

[6]We employ the notion of 'technical' product to terminologically stress its engineering nature (see [BFG$^+$14, Kro12] for a similar terminology). We use the term 'product' for conciseness.

[7]Philosophers usually talk of *being an artefact* as an historical property [Gal98, Dip93] that an object satisfies because of its history. In this sense, one cannot know whether an object is an artefact only by looking at its physical layout; rather, one has to know that it is the result of some intentional production.

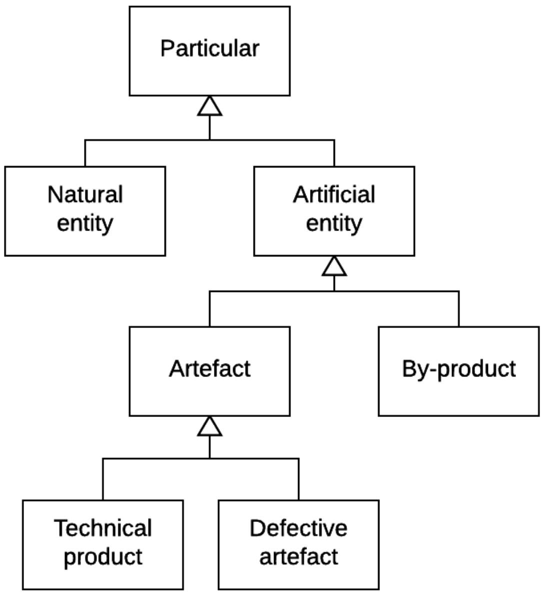


Figure 4.1: Taxonomy of natural and artificial objects

4.2 Design properties

We just saw that a technical product complies (at some degree) with a structured list of design properties. These are differently called across the literature; for instance, some talk about product *designs*, others about product *descriptions*, product *models* or product *types* [TCM13, Gal98, AS09, BZSL10]. Additionally, at the current state of ontology engineering, design properties are not well understood, nor there exists a stable methodology for their representation (see [Sect. 4.3]).

The Industry Foundation Classes (IFC) [Int16] adopts a modelling approach that distinguishes between the classes *IfcProduct* and *IfcTypeProduct* to model the distinction between products and the properties they are required to satisfy. The latter is informally defined as "[...] the specific product information that is common to all *occurrences* of that product *type*" (emphasis is ours). The distinction between type-like (e.g., *IfcTypeProduct*) and occurrence-like (e.g., *IfcProduct*) classes is widely used across the standard; it applies, e.g., to *IfcEvent* and *IfcEventType*, or *IfcDoor* and *IfcDoorType*, among others. Type and occurrence classes are linked via the (reified) relationship *IfcRelDefinesByType*.

An example based on IFC is shown in the figure below (Fig. 4.2). According to the IFC taxonomy, *IfcProduct* and *IfcTypeProduct* are subsumed by *IfcObject* and *IfcTypeObject*, respectively. For the purposes of the example, both *IfcProduct* and *IfcTypeProduct* are specialised in *OccCar* and *TypeCar*.[8] The latter is used to specify, from a general perspective, the properties that instances of *OccCar* have to satisfy. For example, *TypeCar* may include the properties *having_engine_size* and *having_bodywork_dimension*, among others. *TypeCar* is thus the list of properties:

$$TypeCar = \{having\_bodywork\_dimension, having\_engine\_size\}.$$

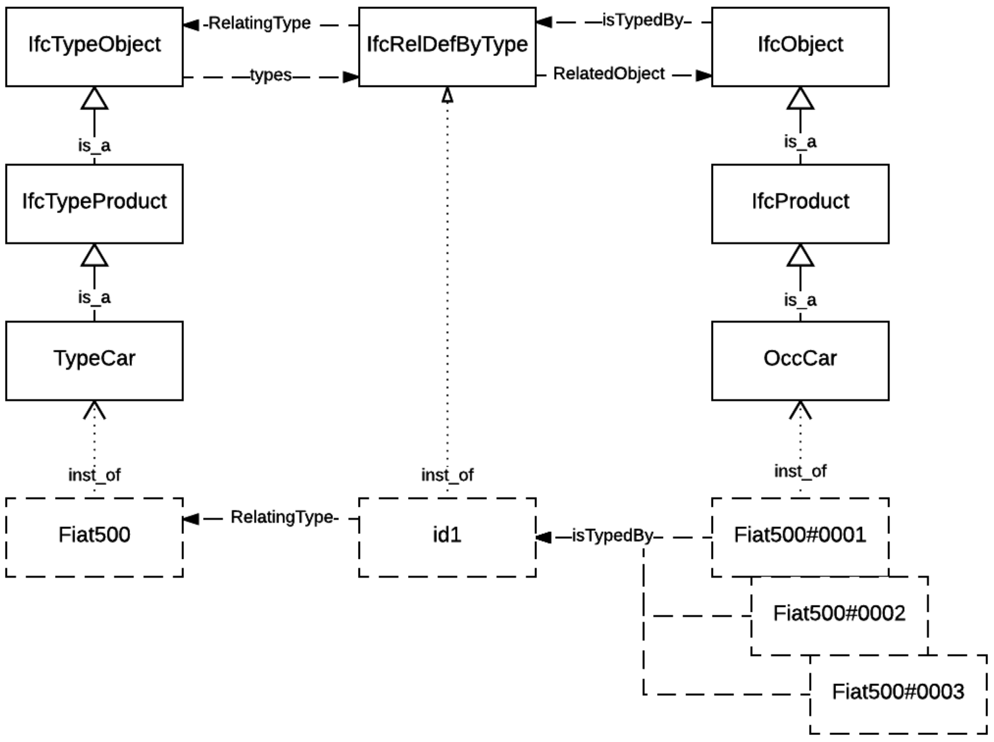


Figure 4.2: Example of IFC (based on [BSŠT14, BSŠT15])

At the instance level, Fig. 4.2 shows some individual automobiles (e.g., *Fiat500#001*, *Fiat500#002*, etc.) (bottom right of the figure) and their corresponding type-instance, *Fiat500* (bottom left).[9] Accordingly, *Fiat500#001*, which instantiates *OccCar*, is the individual car that one owns and drives; it satisfies the properties modelled with specific values by *Fiat500*, which instantiates *TypeCar*, namely:

[8]These classes do not belong to the standard; therefore, they are not labeled with the *Ifc-* prefix.

[9]In this case *Fiat500* is unique for different particular cars. However, IFC practitioners also use different instances for the same type-class to model different properties of occurrence-class instances.

$$Fiat500 = \{having\_engine\_size : 1248cm^3, \\ having\_bodywork\_dimension : 355/163/149cm\}.$$

Along the lines of IFC, Bock and colleagues [BZSL10] distinguish between products and their design properties, which the authors call product models. Accordingly, *Product model* is a UML meta-class having (the subclasses of) the class *Product* as instance(s). Then, *Product* classifies individuals like John's car or my computer, whereas *Product model* describes "[e]xisting or potentially existing things". Additionally, the authors add that "[p]roduct models do not describe documents or other engineering data recording requirements and design [...]".

A similar approach, in this case on temporal entities, is employed in PSL (ISO18629) [Grü09] where the class *Activity* is distinguished from the class *Activity occurrence*. In PSL an activity-occurrence is an event occurring at a certain time and satisfying the constraints established by a corresponding activity. The very same activity can constrain several activity-occurrences since it is meant as "[...] a repeatable pattern of behaviour" [Grü09].[10] *Activity* thus provides the properties (e.g., ordering constraints) that some activity-occurrences have to satisfy.

For example, the activity *hole_making* can constrain the happening of some occurrences of the activities *spot_drilling*, *drilling* and *reaming*, occurring in this order. In this sense, *hole_making* is a complex activity having *spot_drilling*, *drilling* and *reaming* as sub-activities. The PSL-based formula below[11] says that for each activity occurrence x of *hole_making* there is an activity occurrence y of *spot_drilling*, z of *drilling* and w of *reaming*, such that y precedes both z and w, and z precedes w. Recall that in PSL $occ\_of(x, y)$ is read as "x is an activity occurrence of y", and $min\_precedes(x, y, z)$ as "the activity occurrence x precedes the activity occurrence y in the activity tree for z"; see [Grü09] for an overview of PSL.

$$\forall x\ occ\_of(x, hole\_making) \rightarrow \exists yzw\ (occ\_of(y, spot\_drilling) \wedge \\ occ\_of(z, drilling) \wedge occ\_of(w, reaming) \wedge min\_precedes(y, z, hole\_making) \wedge \\ min\_precedes(y, w, hole\_making) \wedge min\_precedes(z, w, hole\_making))$$

Discussion. The modelling approaches above distinguish between particulars (objects or events) and the design properties they are required to satisfy, although the distinction lacks clear ontological foundations and in some case it is not treated in a systematic manner. Instances of *IfcProduct* are material objects, whereas it is not clear what instances of *IfcTypeProduct* are, e.g., whether they exist in space and/or time. Similarly, PSL *Activity*

[10]The reader should not confuse the meaning of 'occurrence' in PSL and IFC; in the latter standard, occurrence is a non-technical term used to refer to some classes; in the former the term refers to particular events *occurring* in time, as it is common in formal ontologies.

[11]The example is based on [Grü09].

occurrence is a class of events occurring in time, but the ontology leaves unspecified what it means for an instance of *Activity* to be “a repeatable pattern of behaviour”. Interestingly, Bock and colleagues [BZSL10] stress that in their approach a product model is not a document; what it is then remains however unclear.

From a formal perspective, PSL represents activities via reification. As briefly mentioned in [Sect. 3.5], reification is a technique used in logic and knowledge representation that allows us to quantify over predicates in a first-order setting. In the PSL formula above, *hole_making* is an example of reified predicate, where $occ\_of(x, hole\_making)$ is the *instance of*-like relationship used to say that x instantiates *hole_making*. In the case of IFC, type-like classes can have instances, therefore also in this case properties are treated as individuals. However, IFC is not represented in formal logic, therefore reification is not technically specified. Instances of the reified relationship *IfcRelDefinesByType* may be understood as providing an instantiation constraint. Bock et al. [BZSL10] employ a different modelling approach based on UML meta-classes, by which *Product model* is a second-order class having the first-order class *Product* as instance (see also [PZMY94, DPZ02]). If the approach were to be represented in formal logic, reification would be a suitable technique to avoid the use of a high-order logic and to stick on a first order setting. However, reification as formal technique is of no help in understanding the ontological nature of the entities being represented. We shall now propose how design properties can be understood from an ontological perspective.

4.2.1 Product types as concepts

In order to address the distinction between products and design properties, we find useful the adoption of the *semiotic triangle* as developed in semiotics. Recall that in the triangle the word ‘red’ in its predicative sense is related to the concept of *being red* (its *intension*) and to the class of things that are red (its *extension*) (see Fig. 4.3). Different interpretations of the semiotic triangle are proposed [GV99]; we rely on the notions of intensionality and extensionality, because of their use in logics and applied ontology.[12]

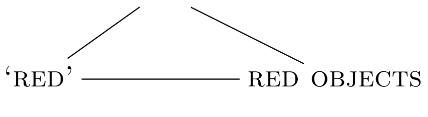


Figure 4.3: Semiotic triangle

[12]Our use of the semiotic triangle is not meant as a proposal for a new interpretation of the triangle itself; as said, we find it helpful to present our approach.

By means of the semiotic triangle we distinguish between (see Fig.4.4):

- *Technical product type* (product type for simplicity), a structured list of properties that a particular has to satisfy (at some degree) to be a technical product; for example, ‘Fiat500’ or ‘Iphone6’ are product types;
- *Technical product*, a produced particular that complies (at some degree) with a product type (see [Sect.4.1]);
- *Technical specification* (specification), the encoding of a product type in a (representational) language.

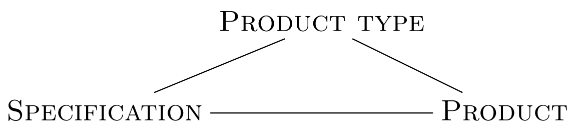


Figure 4.4: Semiotic triangle for design

Some clarification are due. First, product types are not identifiable with products. This view copes with the fact that design properties may be about not yet realised physical items. Practitioners can interact about the product to be realised, even if there is no physical counterpart of the properties. In principle, some product types may be created even if their corresponding products are never produced.

Second, a product type is not a specification, the latter being a *representation* of the former in a technical language. For example, the product type *Fiat500* is a (structured) list of properties whose specification may be provided in a graphical CAD language or in plain English. Also, a specification is not to be confused with the *technical support* where the specification is concretised, e.g., a blueprint, a paper-made catalogue or a computer pdf file. Supports are physical objects in space and time, which are sometimes constituted by material.

We shall thus treat product types as having an *intensional nature*, namely as being independent from their extensions, i.e., the particular entities they classify. After the work of Carnap, intensional properties are usually approached in logic from a modal perspective, i.e., as functions from possible words to domain of entities. On our side, we shall pursue a different approach based on the theory of concepts presented in [Sect.3.5] and further developed in [Ch. 6]. We assume that a product type is created and can be possibly destroyed; it is therefore some sort of particular that can classify some physical objects, in the sense that the latter are its instances. However, differently from these, a product type is not *per se* located in the physical space. Also, we share Galle's remark [Gal98] in that a modal framework does not make justice of the way designers think about their daily work, since they do not describe products in terms of "[...] possible worlds other than the actual world in which [they] [live]" [Gal98, p.71].

Finally, note that when designers talk about product *models*, e.g., assembly or geometric models, the distinction we have just drawn between product type, specification and support is blurred. Therefore, to talk about, e.g., the *model* of a car, one should distinguish the properties that the car satisfies from the encoding of the properties in a language and their concretisation in, e.g., a paper-made catalogue.[13]

4.3 Comparison with the literature

In the following we compare our notions of product and product type with some relevant works across engineering, applied ontology and philosophy.

Products. At the current state of art there is a proliferation of ontologies for product knowledge representation, given their increasing application in advanced modelling systems. However, most of these ontologies are poorly axiomatised and the motivations behind the formalisms are hardly provided, as we saw across [Ch. 2]. We shall here compare our approach with the analysis presented in [BFG$^+$11, BFG$^+$14], where the authors provide a unified perspective about artefacts. The purpose is to contextualise our approach within the larger framework they provided.

The work of Borgo and colleagues [BFG$^+$11, BFG$^+$14] aims at integrating, via ontological methods, three different perspectives on technical artefacts embraced in applied ontology (the *ontological artefact* view), engineering design (the *engineering artefact* view) and philosophy of technology (the *technological artefact* view). The authors argue at length the principles motivating their perspectives and adopt DOLCE as common foundational theory for the comparison of the proposed views. The authors propose the following three definitions for the notion of technical artefact on the grounds of the three aforementioned perspectives:[14]

Ontological Artefact view (OA): "A technical artefact a is a physical object which an agent (or group of agents) creates by two, possibly concurrent, intentional acts: the selection of a material entity (as the only constituent of a) and the attribution to a of a technical quality" (see also [BV09]);

[13]Admittedly, a product *model* may be also understood as a physical prototype, i.e., a physical object that resembles (at some extent) the final product and is used by experts, e.g., for testing purposes. The ontological characterisation of prototypes is behind the purposes of our work.

[14]The following quotations are taken from [BFG$^+$14].

Engineering Artefact view (EA): "A technical artefact *a* is a physical object which an agent (or group of agents) creates by an intentional act: the carrying out of a production process with the goal of obtaining a physical object *a* that is expected to realise a given behaviour or a given property when participating in given generic technical situations";

Technological Artefact view (TA): "A technical artefact *a* is a physical object which is, firstly, created by the carrying out by an agent (or group of agents) of a make plan for a physical object with a physical description *id* (which includes performing a checking procedure that establishes that the object sufficiently answers to the description *id*) and for which, secondly, a use plan exists" (see also [HV09]).

The OA is the most general view; accordingly, a technical artefact is not necessarily the outcome of a production process by which the physical properties of an object are intentionally changed to meet design properties. Rather, an OA exists because a physical object is *cognitively selected* and *ascribed with* a technical quality. For instance, a pebble is selected and ascribed with the ability of performing as a paperweight.[15]

The EA prescribes for a technical artefact to be the result of an intentionally performed production process, whose purpose is to realise a physical object that is able to exhibit a specific behaviour under a given situation. For instance, a screwdriver is an EA if and only if (*i*) it is the outcome of an intentional production process and (*ii*) it is able to realise the behaviour for which it was designed. Additionally, a physical object that is defectively produced may still be an EA as long as it can manifest the behaviour it was intended to exhibit.

Finally, a TA is the outcome of a production process that is intentionally realised on the basis of a production plan (*make plan* in the authors' terminology), which is carried out to realise a certain design specification (*physical description*). Thus, a produced object has to satisfy its corresponding design (at the physical properties level) to be a TA, and the compliance has to be checked. Additionally, this latter view emphasises the role of use plans for TAs, where a use plan describes how a technological artefact has to be used to realise a goal.

We share with [BFG$^+$14] the core idea that technical products (*technical artefacts*) are intentionally created. Differently from the OA view, we share with the TA and EA views the overall understanding of technical products as particulars that are the outcomes of production activities

[15]The notion of 'technical quality' is not defined in [BFG$^+$14] and is borrowed from [Dip93]. Informally, it is a quality that an agent attributes to an object and that is meant to grasp why the object is useful for the agent.

by which their physical properties are manipulated to meet design properties. The OA strives, indeed, from a more general and foundational layer, whereas our focus is on design and manufacturing. Our notion of technical product can thus be subsumed under OA, when an agent intentionally selects an object via physical manipulation. Differently from the EA, we hold a more relaxed view on functionalities, in the sense that it is up to experts to decide which design properties an object has to necessarily satisfy to comply with its corresponding product type. From this perspective, our notion of technical product is more general than the EA as we cover compliance with design properties, the latter being generally understood. Our approach allows designers to constrain to functionalities the essential properties that an object has to satisfy to be a technical product. Therefore, in our framework an instance of EA is a technical product that complies at least with the functional properties of its product type. Finally, we hold a more general perspective than the TA for at least three reasons: first, we do not prescribe the existence of make plans. Accordingly, a product type (or its specification) may or not include production constraints; in our view, it is up to expert to decide whether the latter are mandatory or not. Second, in our view checking procedures provide engineering relevant information about the compliance status of the checked object. Differently, according to the TA view, a check affects a TA's identity in the sense that an object is not a technological product if it does not pass a check. Third, we do not force a technical product to be released with a use plan; an object is a technical product because it is produced in compliance with its product type. If it is used in a way that is not supported in the corresponding type (e.g., a chair as door-stopper), the product can play a technical role, but the (mis-)use does not affect the product identity. A TA can thus be seen in our approach as a technical product that is released with a use plan, passes a checking procedure and has a product type (or specification) which includes manufacturing constraints.

Product types. In the philosophical analysis of design and technology, product types (designs, descriptions, etc.) are not well understood. Kroes [Kro12], for example, states that: "When referring to a car *design* [...] what is meant is [...] something that has more to do with the *properties* of the car itself, irrespective of whether the car actually exists or how (if it indeed exists) it was actually produced. It is not easy to grasp what this 'something' is" [Kro12, p.146] (emphasis is ours).

Hilpinen [Hil93] provides an artefact theory according to which a person designs a product under some *description* that defines the product's intended properties: "When a person intends to make an object, the content of the intention is not the object itself, but some description of an object or some 'concept' under which the intended object is conceived; the agents intends to make an object of a certain kind or type [...] which the artifact should exemplify" [ibid., pp.157–58]. Accordingly, a description is not a written document, but a concept which describes the properties that some objects have to satisfy. Also, Hilpinen distinguishes *type descriptions* (also called *sortal descriptions*) from *adjectival descriptions*. The former "determines the identity of an object and the criteria by which it can be distinguished from other objects" [ibid., p.158], whereas the latter describes what Hilpinen calls the "character" of an artefact, namely the properties that do not affect an artefact identity, e.g., qualities. To use his examples, *house* and *shirt* are type descriptions, whereas *red* is an adjectival description.

In different papers, Galle [Gal97, Gal98, Gal08] provided a systematic analysis of design representations in terms of what he calls *design objects*, namely the (generally speaking) entities that are at the focus of designing practises. The starting point of Galle's investigation is the consideration that during a designing activity, designers talk as if there were a physical artefact in the world, e.g., talking about its qualities, when actually there is no such an artefact. For example, a designer may say that the building she designed complies with some regulations about fire safety and energy consumption, even though there is no physical building at the time she speaks. What is then the building (a design object indeed)? This is what Galle calls *the problem of the absent artefact* [Gal98]. In [Gal97, Gal98], he proposes looking at design objects as *artefact ideas* "in the sense of states or objects of the mind, or non-material entities accessible through cognition" [Gal98, p.72]. From this perspective, recalling the example above, it is the idea of the building that complies with some regulation, as Galle himself recognises [ibid., p.80]. Then, a design representation is a material object that a designer produces driven by her artefact-ideas. More recently, Galle [Gal08] presents different interpretations for design objects. Although his purpose is to provide a method for the conceptual analysis of design, rather than giving the last world on the problem of the absent artefact, he seems to put forward the claim that design objects are "abstract entities [...] existing but not in space-time" [ibid., p.290]. He recalls Thomasson's [Tho99] understanding of abstracts as entities that can be created and destroyed, despite of their lack of spatial-temporal location. In a different interpretation, Galle recalls Alexander's [Ale79] view concerning *architecture objects* (i.e., design objects within architecture) understood as complex patterns of relationships between their composing elements. For example, a church aisle (as architecture object) is a "pattern of relationships between its length, its width, the columns which lie on the boundary with the nave, the windows which lie on the other boundary [...]" (quoted in [Gal08, p.284]).

To the best of our knowledge, Galle provided the most systematic analysis of product types (design objects) assessing the pros and cons of their interpretation according to different philosophical theories. As himself concludes in [Gal08], these interpretations require however to be evaluated against their "theoretical relevance to design" to understand if and how they are suitable to capture design practises. The analysis of product types given in [Sect. 4.2] is on the one hand in line with some of the interpretations given in [Gal08], while on the other hand it is grounded on the specialised literature to be relevant for modelling purposes. Additionally, we distinguish between different entities (i.e., product types, specification, physical support), whereas these distinctions are only vaguely addressed by Galle.

In applied ontology the TA view in [BFG$^+$14] assumes that a particular is a technological artefact if it satisfies "a physical description *id*", among other things. Its proponents also argue that it is a description that determines which type of technical artefact a particular is. However, they do not provide an explanation for descriptions.

In the analysis of technical artefacts given in [Gua14], Guarino argues: "[...] I do not require a design specification to be explicitly encoded on a publicly accessible physical substratum: it could also be encoded in the mind of an agent [...]". Recently, Guarino and Stufano [GSM15] propose to understand a product type as some sort of "virtual object" that lacks spatial location. They claim[16] that a virtual object is some sort of fictional entity that exists because of experts' talks and that can be possibly realised in a material object. Their approach is however preliminary. The authors should clarify (at least) what a virtual object is, as well as its relationship with physical products. In [Appendix A] we compare Guarino's approach and our theory of technical products with respect to the so-called *missing and replaceable artefact problem.*

The Information Artifact Ontology (IAO) [SMR$^+$13, SC15] is an ontology for the representation of documents, mainly used to represent health care records. It has been developed by specialising the Basic Formal Ontology (BFO) [ASS15]; Fig. 4.5 shows the core classes of the IAO as specialisation of BFO.

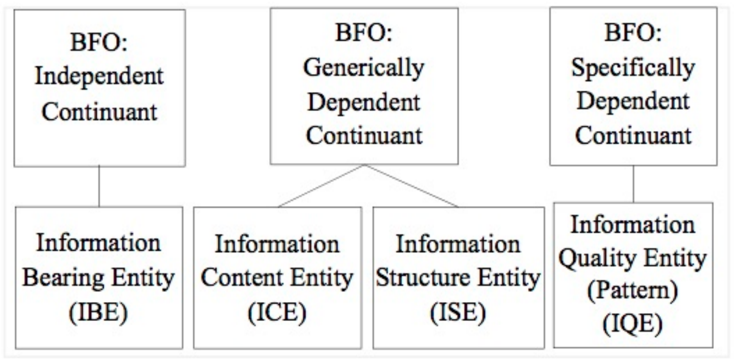

BFO: Independent Continuant
BFO: Generically Dependent Continuant
BFO: Specifically Dependent Continuant
Information Bearing Entity (IBE)
Information Content Entity (ICE)
Information Structure Entity (ISE)
Information Quality Entity (Pattern) (IQE)

Figure 4.5: IAO core classes (from [SMR$^+$13]).

[16]Personal communication.

An *information content entity* (ICE) is an generically dependent continuant that is "about something in reality".[17] As a continuant, an ICE is a particular that exists in space and preserves its identity through time. It is generically dependent, in the sense that whenever it exists, there exists some object upon which it relies. An example of ICE is the "content" (also called *information* in [SMR+13, SC15]) of John's health record, which is about John's health status and generically depends on some computer file, or piece of paper. These latter two are called *information bearing entity* (IBE) and are material objects that bear information. An *information quality entity* (IQE) is "the pattern on an IBE in virtue of which it is a bearer of some information". An example of IQE is an ink mark in a book; it is a specifically dependent continuant, because it depends on a very specific IBE. According to the IAO, an information content entity is *concretised* by an information quality entity of some information bearing entity. The following quotation provides a clarification: "A journal article is an information [content entity] that inheres in some number of printed journals. For each copy of the printed journal there is some quality that carries the journal article, such as a pattern of ink. The quality (a specifically dependent continuant) concretizes the journal article (a generically dependent continuant), and both depend on that copy of the printed journal (an independent continuant)" (quoted in [Gar16]). Finally, an *information structure entity* (ISE) "is a structural part of an ICE; speaking metaphorically, it is an ICE with the content removed: for example an empty cell in a spreadsheet; a blank Microsoft World file".

We share with the IAO the overall idea of distinguishing between a physical support and what is specified in the support by means of a language. Differently from our approach, however, the IAO sticks on the ambiguous notion of information (*information content entity*). This is conceived as a continuant that generically depends on some object and, according to some agent, is always about some portion of reality.[18] Recall that according to [SC15], if information x is *about* y, then y exists, i.e., it is physically present. On our side, what we call product type does not always classify physical products, since it can describe not yet produced products. From this perspective, product types do not correspond to information content entities in the IAO sense.

[17]Quotations concerning the IAO are taken from [SMR+13] if not otherwise specified.

[18]The ontological nature of information is highly debated in the literature [Adr13], whereas the IAO does not attempt any comparison with current existing theories of information.

Chapter 5

Engineering features

In this chapter we provide an analysis of engineering features, which – as we saw in [Ch. 2] – are core elements to model product knowledge. By looking at how engineering features are used and understood across the literature, we individuate from a general perspective the entities that they are meant to model [Sect. 5.1]. This will bring us into the high-level distinction between feature types and physical features [Sect. 5.2], as well as into the analysis of ontological features [Sect. 5.3] and components [Sect. 5.4]. By the end of the chapter we look at form features [Sect. 5.5], because of their relevance for product modelling.

As a remark, the terminology is unfortunate; we talk of 'ontological feature' with reference to ontological dependent objects [Sect. 5.3] and 'engineering features' with reference to the notion of feature as used across the product modelling literature. We say more about the relation between engineering and ontological features throughout the chapter.

5.1 What do engineering features represent?

Engineering features have been understood from the very beginning of feature-based modelling approaches in tight connection with cognitive phenomena and the need of abstracting from pure geometry in the practise of product modelling. 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 [...]" [SM95, p.13]. The authors also stress that a feature is a "significant" element for a product lifecycle management task [SM95, p.97], namely, something that has to be taken into account for some application purpose. This view is recalled in various definitions proposed over the years (see [Ch. 2], Table 2.1); e.g., a feature as "the engineering significance of the geometry of a part" [Win91] or "a region of interest in a part model" [BJ93].[1] More explicitly, Rossignac [Ros90] introduces the notion of *intentional feature* as "an abstraction of geometric elements", by stressing terminologically that a feature is *intentionally* created and used for certain design purposes.

As stated earlier, features are used in a variety of ways to model disparate entities. However, not everything is a feature; a whole product, for example, is never considered a feature; it is rather said to be an "aggregation of features" [SM95, YJ06, Chu10]. The implicit idea seems to be that features model (generally speaking) entities that "characterise" products and exist as far as products exist. More specifically, from the review of the literature [SB16], it emerges that – from a high-level perspective – (engineering) features are employed to model (see Fig. 5.1):

- **Ontological features** like holes, slots and bumps;
- **Qualities** like shapes, dimensions and weights;
- **Materials** like wood and plastic;
- **Components** like screws and building walls.

Ontological features are the main ontological entities modelled in feature-based approaches. For example, form features model ontological features (see [Sect. 5.3]) along with their shapes;[2] machining features attach machining knowledge to ontological features, e.g., in order to specify the resources necessary for creating a slot on a workpiece; functional features ascribe functionalities to ontological features, e.g., a hole for assembly purposes.

Qualities are used to specify products' characteristics, shapes and dimensions foremost, along with their quantitative values. Qualities are also used to characterise form features; e.g., a hole may be ascribed with a cylindrical shape and a nominal diameter of 12mm with a tolerance of $\pm$ 0.8mm.

The representation of materials is adopted to specify the amount of matter constituting a product, a workpiece or a form feature like a bump. Material information is relevant in various phases of product lifecycle; e.g., it is necessary to know which material constitutes a workpiece to select the proper tools for machining.

The representation of components in terms of (engineering) features is also found across the literature, despite it is less common than ontological features or qualities. In feature-based approaches components are sometimes called *structural features* [RRB15].

[1]Recall that 'region' is commonly used in product modelling with the meaning of product's part [YB92].

[2]To be precise, form features are differently used with reference to various entities; we dig into the analysis of form features in [Sect. 5.5].

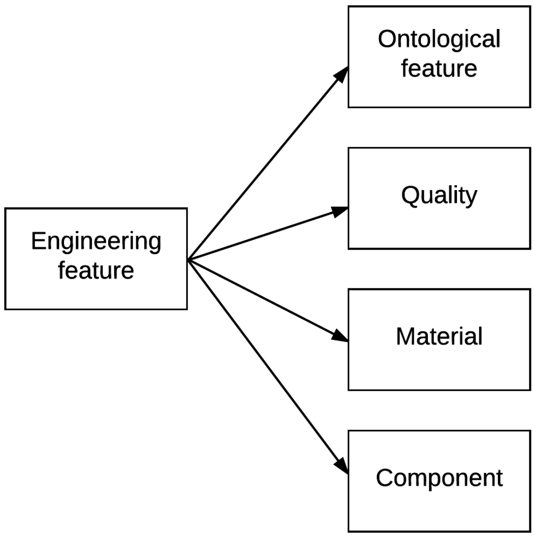


Figure 5.1: Entities represented by engineering features

The variety of entities modelled in feature-based approaches is not surprising; recall that it is common to look at features as anything being relevant for the modelling task at hand [UYC+13]. From an ontological perspective, it is important to individuate what domain entities are represented by engineering features, because of the differences in the nature of these entities and, consequently, the different ways in which they relate to other entities. For instance, the relationship that a product holds with a shape quality differs from the one it holds with a hole. Differently from a quality, indeed, a hole is not spatially included in a product, at least if holes are understood as immaterial objects (see [Sect. 5.3]).

As we saw in [Sec. 2.2.3], most of the data modelling resources used to handle feature-based data adopt underspecified relationships between material objects and features. In our approach, once the entities modelled by engineering features are identified (see Fig. 5.1), the high-level relationships with material objects, products foremost, can be easily derived. To be more precise, when a product is loosely said to "have" an engineering feature, the following relationships have to be distinguished:

- **Hosting**, when engineering features model ontological features, e.g., a hole or a bump. As previously said, ontological features are the main entities represented in feature-based approaches. The *hosting* relationship thus plays a fundamental role. Informally, when a product *hosts*, e.g., a hole, the latter cannot be present in time-space without the former. In this sense, *hosting* establishes the existential dependence of an ontological feature on its *specific* host (see [Sect. 6.2]);

- **Inherence** (see [Sect. 3.4]), when engineering features model qualities, e.g., shapes. Accordingly, a product has a form feature means that there is some individual quality of the shape kind that *inheres in* the product. As we saw in [Sect. 3.4], individual qualities in DOLCE-CORE are organised into quality spaces. For product knowledge representation, quality spaces are useful to represent quantitative values attached to qualities, e.g., dimension values;
- **Parthood** (see [Sect. 3.3]), when engineering features model products' components. *Parthood* is indeed the most common relationship employed – at the high-level – for assembly representation [FGL+07, KYK08, RHF+06]; although, more specific parthood-like relations may be needed when dealing with specific assembly knowledge;[3]
- **Constitution** (see [Sect. 3.7]), when engineering features are used to specify the materials making up a product.

Finally, note that the entities modelled by engineering features are represented as both physical particulars and design properties. We explore this distinction in the next section.

5.2 Feature types and physical features

In the previous chapter we stress the distinction between physical objects and the design properties that they are required to satisfy in order to be considered as technical products (see Fig. 4.4 in [Sect. 4.2]). The same reasoning applies to engineering features. As a design property, a feature like a hole is not a physical entity and it only exists within a product type, where the hole design property can be characterised, e.g., in terms of shape and dimension. On the other hand, a physical hole in a product can be considered as the realisation of the hole design property if it is created to comply with such a property.

We thus propose the following distinction between *feature types* and *physical features*:

Feature type A design property used in a product type to represent an "aspect" of the product type. The intended meaning of design feature is captured in the literature by definitions like: "An information unit describing an aggregation of properties of a product model that are relevant in the scope of a specific view on the product" [Den99], "a set

[3]An initial investigation of *parthood*-like relationships for assembly is presented in [SMBP16].

of information related to a part's description", "a region of interest in a part model" (the latter two are both quoted in [BJ93]);

Physical feature A physical entity that is related to a physical product and that satisfies a feature type. This reading is captured by definitions like: "The characteristics of a product that result from design" [Gro07], or "[...] a physical constituent of a part" [SM95].

As said, the two notions are strictly related: physical features are the physical entities that instantiate feature types, e.g., a physical hole feature in a gear that instantiates a hole feature type. Note that not all feature types are instantiated, e.g., because they are introduced as design alternatives or because the product type to which they are related is never physically produced. Despite their relation, the notions of feature type and physical feature cannot be confused: physical features are in space and time, whereas feature types only exist within product types, therefore they are not in space. Additionally, both feature types and physical features cannot exist in isolation. This means that an engineering feature is ultimately related to a product type or product.

These considerations suggest that the notion of feature in engineering has a relational and dependent nature. This is a shared view across the literature, even though it is not always represented. In IFC, for example, the class *IfcFeatureElement* is defined as "the generalization of *existence dependent elements*" (emphasis is ours). In the Design Ontology [ŠAM10], as we saw in [Ch. 2], the class *form feature* is subsumed by *material object*, meaning that its instances cannot exist without being related to material objects. Explicit dependence links between features and products are represented in [TCM13, VHB00], although their semantics is not formally given.

The distinction between feature types and physical features has been existing for a long time, although it has been blurred and not treated in a systematic manner. Salomons and colleagues [SVHK93], for example, refer to feature types when claiming that a feature is "a carrier of product information that may aid design or communication between design and manufacturing, or between other engineering tasks". Similarly, in their seminal work Shah and Mäntylä [SM95] see a feature as an "information cluster" for integrated product representation. The authors, however, muddle the distinction when adding that "a feature is a *physical* constituent of a part" [SM95, p.97] (emphasis is ours).

To sum up, at the type level a feature is a (possibly complex) design property related to a product type;[4] e.g., a hole feature type that characterises a gear product type, such that each instance of the latter has a

[4]In the next chapter we provide a systematic treatment of the relationships holding between feature types and product types.

physical hole with a certain shape and diameter. Generalising, physical features are physical particulars (ultimately) related to products that comply (at some degree) with corresponding feature types.

In the remaining part of the chapter, we analyse (some of) the entities being modelled by engineering features. The notions of quality and material (amount of matter) were already introduced in [Ch. 3] (see [Sect. 3.4] and [Sect. 3.7], respectively). We now investigate how ontological features and components can be understood in a way that reflects engineering knowledge. By the end of the chapter we look at form features, given their relevance in feature-based modelling approaches.

5.3 Ontological features: An informal overview

As said in the previous sections, ontological features are amongst the most common entities represented in feature modelling approaches. Despite this, their ontological nature is neither analysed, nor formally represented in ontologies for product representation.

In [Sect. 3.7] we mentioned that ontological features in DOLCE are particulars that exist as long as they are ultimately related to some non-feature object. The ontological nature of these entities has raised some interest in philosophy, although at the current state there is no unified theory that covers surfaces, holes or bumps, among other ontological features. Stroll [Str88], for instance, proposes an ontological analysis of surfaces, whereas [LL70, CV94] have stimulated the debate on the nature of holes.

Taking the latter as paradigmatic examples of ontological features, deflationary philosophical theories hold that holes are not objects on the par of, e.g., desks or cars. A hole is an object quality [Mea15]; e.g., a holed (physical) gear is a gear with a concave shape but there is no hole-object in the gear. On the other side, on the grounds of a common sense understanding of reality, also supported by cognitive studies [GB00, BC14], inflationary theories claim that holes exist as dependent objects [CV94].[5] In this perspective, a holed gear is said to *host* a hole-object, where – as said – the *hosting* relationship establishes that the latter cannot be present in time without the former.

Philosophers that agree on the existence of holes as objects, however, disagree about what holes are, e.g., whether they are material [LL70] or immaterial [CV94]. In the latter case, a hole is a void space in an object, while in the former it is an object material surface with a concave shape.[6]

For the purposes of our work, ontological features like bumps need to

[5]The distinction between deflationary and inflationary theories of holes is borrowed from [Mil07].

[6]In the philosophical debate about holes, there are different variations of these positions; see [CV14] for an overview.

be taken into account as well, because of their use for product modelling. As in the case of holes, one may assume either that a bump is an object on its own, or that it is a shape quality that characterises a certain object. In the former case, bumps are doubtless constituted by material. In this view, a bump may be seen as an object's part that is therefore located within the region where the whole object locates. Additionally, along with holes, a bump cannot exist if not related to some other object. In this sense, it can be understood as a *material ontological feature.*

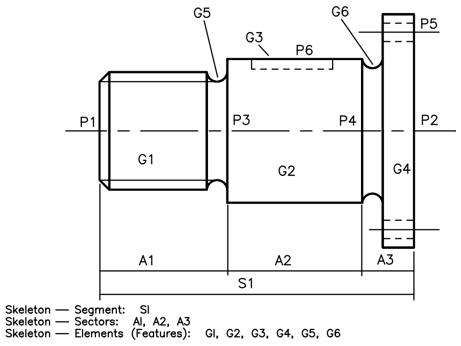
Discussion. As regards the ontological nature of holes, we will see in [Sect. 5.5] that the philosophical views concerning whether they are shape qualities, material or immaterial (dependent) objects are all found in engineering for form feature modelling. Also, we will see by the end of the chapter [Sect. 5.6] that in formal ontologies holes are commonly treated as immaterial objects. For the time being, let us recall that ontologies taking holes in the immaterial sense do not treat them as objects' parts.[7] The reason is that when mereology is used for parthood relationships between physical objects, there is a bridging relation between the properties *being part of* and *being spatially located in* [CV99]. Consider, e.g., a car's bodywork: the bodywork – as part of the car – locates in a space region within the (larger) region of the car. Differently, a region where a hole locates is not within the region where its host locates. Rather, a hole locates in the *convex hull* of the host, namely in the smallest convex region occupied by the host [HB12]. We will go back to this in [Sect. 5.6].

[7]'Part' is here used in the mereological sense (see [Sect. 3.3]).

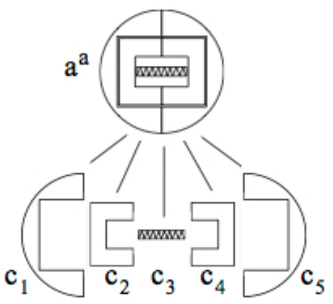
5.4 Components

Most technical products consist in the (structured) aggregation of various components. These can be either *constructional parts* or *assemblies*. The former form the lowest level of the structural hierarchy in the sense that they "[...] are not normally capable of further disassembly" [EH08, p.4] without being destroyed. Examples include screws and resistors. The latter, as the name suggests, result from the composition of components, which in turn can be constructional parts or (simpler) assemblies. Products like airbuses, laptops and chairs are some examples (see [Whi04] for assembly design).

Constructional parts and assemblies are differently represented. Graphical models of the latter usually specify (although in different manners) the components making up an assembly [IY15, DFEG$^+$90, Boo94] whereas this is not done for the former, see Fig. 5.2.


G5
G6
G3
P6
P5
P1
P3
P4
P2
G1
G2
G4
A1
A2
A3
S1
Skeleton — Segment: Sl
Skeleton — Sectors: Al, A2, A3
Skeleton — Elements (Features): Gl, G2, G3, G4, G5, G6

(a) Constructional part (from [EH08])



(b) Assembly (from [SMBP16])

Figure 5.2: Examples of graphical representations of components

In ontological terms a component can be understood as a particular that is intentionally designed to be assembled. For example, a car engine is a component, because it is designed to be assembled to a car, whereas the car is not a component, because it is not designed to be assembled to anything else. As we will see, this is a common view across the literature and international standards. However, two questions arise, namely (*i*), whether it is a necessary condition for a component to be a technical product, (*ii*) whether it is necessary for a particular to be assembled to some other object in order to be a component.

Regarding the first issue, a stone (natural object), for example, may well be used to build a wall; we may thus think of the stone as a wall's component without inferring for the stone to be a technical product.[8] Data modelling resources adopt different perspectives on this point. The STEP standard

[8]We assume in this example that the stone is not physically modified, i.e., it is used in the wall in the same shape, material volume, etc., as it was before its usage in the wall.

AP224 application protocol, for example, holds that the class *Component* "[...] specifies either a single piece part, or another manufactured assembly used to define an assembly" [fSI06, §4.2.128.2]. In the same standard, the class *Part* refers to "[...] the physical item which is intended to be produced through the manufacturing process" [fSI06, §4.2.155]. It is thus assumed for a component to be a technical product; then, the stone of our example is not a component of the wall, even though it is amongst its parts. On the other hand, the standard ISO 19439 defines a component as an "[e]ntity that is part of, or capable of becoming part of, a larger whole" (quoted in [UYC$^+$13]). The standard thus seems to assume a more general perspective compared to STEP-AP224, since components are not necessarily products.[9] In this perspective, a stone in a wall is a component.

As regards the second question mentioned above (whether it is necessary for a particular to be assembled to some other object in order to be a component), consider, e.g., a car engine that at time t is not amongst the components of any car; it just lays in a storehouse ready to be used. The question is whether the engine is a component at t. On the one hand, it seems reasonable to think of an object as a component because it is part of another object. For example, we may assume that John's car includes the engine amongst its components at time t, because the engine is indeed part of the car at t. It would be odd to classify the engine as a component at t, if it were not part of John's car at that time. This reading seems to be behind the STEP-AP224.

On the other hand, it is also reasonable to think of a component in virtue of its design properties. An object is, indeed, developed as a component, i.e., it is explicitly designed as such. In this sense, an object is a component at a certain time not because it is part of some other object at that time, rather because it has been designed to be assembled to some other object. This seems the reading of the ISO 19439, where a component *is or is capable of* becoming part of a larger whole, to rephrase the definition quoted above.

Discussion. In our understanding, at least two notions of component have to be distinguished:

1. *Technical component* (simply, component), a produced object that complies (at some degree) with a product type, which establishes that the component may be part of a product at a certain time. A component is thus a technical product.

2. *Component role*, an object that is intentionally used as part of another object.

[9]We draw this conclusion from the generality of the term *entity* in ISO 19439, whereas STEP-AP224 is explicit about the artefactual nature of the entities classified under *Part*.

According to the second definition, a natural object can have a component role. For example, a stone is used as a certain wall's component and the artisan who builds the wall assumes that the stone can bear a certain weight, possibly in virtue of (some of) its qualities like volume and shape. The artisan thus assumes that the stone will not break and, on the contrary, will support the stones (or bricks) on its top.

5.5 Form features

In this section we provide an ontological analysis of form features by looking at how they are understood across the specialised literature. Amongst the various classes of features used in engineering, we focus on form features because of their fundamental relevance.

First, form features are likely the most used engineering features. They are traditionally employed to represent both hole-like and protrusion-like entities. Recall that the former are called *subtractive* and the latter *additive* features [TCM13].[10] Second, different engineering feature classes are form features enriched with application specific knowledge. For instance, machining features attach to form features the relevant details for manufacturing; functional features enrich form features with the functionality they play within a product. In [UYC$^+$13], for example, the *form feature* class subsumes all classes of engineering features.

By reviewing the state of art, we individuate three different ways of conceptualising form features.

In the **first view**, a form feature models an object's shape for which nominal dimensions (and their tolerances) are commonly given [SVHK93, UY14, IY15]. According to Shah and Mäntylä, for example, form features are "recurring, stereotypical shapes" [SM95, p.98]. For instance, a physical hole[11] (or a protrusion) is a product's shape. Hence, according to this view, a sentence like "The product has a hole (protrusion)" has to be rephrased in "The product bears a hole-like shape (protrusion-like shape)". Fig. 5.3 shows (the sections of) two (solid) objects, one with a protrusion-like shape (Fig. 5.3a) and one with a hole-like shape (Fig. 5.3b).

[10]This terminology is due to the fact that physical holes are obtained by 'subtracting' material, whereas physical protrusions by 'adding' materials.

[11]We often use holes as examples of engineering features, because these are mostly discussed in the specialised literature [CY11, BNM08, YJ06, EKK$^+$13].

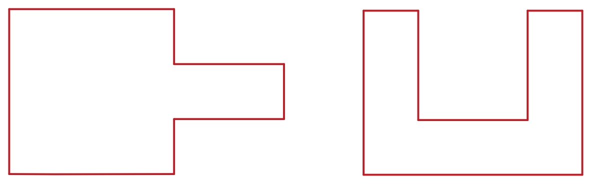

(a) Object with protrusion-shape (b) Object with hole-shape

Figure 5.3: Form feature: first view

In the **second view**, a form feature models an object's part, which bears a characterising shape [AHYC12b, WP88, VHB97]. As previously said, it is indeed common in engineering to talk of form features as shaped products' regions [SVHK93, KWMN04]. Accordingly, a sentence like "The product has a hole (protrusion)" is rephrased in "The product has a holed-part (protrused-part)". This reading is particularly adopted when experts refer to the material properties of form features [EKK$^+$13]. For example, a so-called *reamed hole* is a hole obtained by a reamer and whose material surfaces are smoother than the ones of a so-called *drilled hole*.[12] This perspective on the materiality of holes as objects' parts share some similarity with the philosophical theory proposed in [LL70] and briefly presented in [Sect. 5.3]. Fig. 5.4 shows (a section of) an object with a protrused part (Fig. 5.4a), and (a section of) an object with a holed part (Fig. 5.4b).

Note that differently from the first view, a protrusion (or a hole) here does not reduce to an object's shape; it is rather an object with a characterising shape. From an ontological perspective it is challenging to determine the exact nature of, e.g., a holed-part. Is it an object material boundary (a *hole-lining* in the terminology of [LL70]), or is it rather an extended object part? In the former case, which is the hole-lining to be considered in an object? In Fig. 5.5b, for example, one may consider either the red, or the violet or the orange hole-linings, among others. A similar consideration can be done for protrusion form features; e.g., does a protrused-part extend as in Fig. 5.4a, or does it rather extend through the entire object as in Fig. 5.5a? Different point of views can be assumed for these cases, depending on the ontological understanding of the entities at hand. From an engineering perspective, we leave to experts to decide – within this second view about form features – which is the form feature part in an object to be considered. In our experience, it seems that engineers tend to consider material holes in the sense of hole-linings as in Fig. 5.4b, e.g., because these are the (material) parts that undergo manufacturing operations, or because they can be ascribed with certain design properties. For example, if a drilled holed-part

[12]The use of a reamer allows for the realisation of finer surfaces; see, e.g., `http://www.custompartnet.com/wu/hole-making`, last access October 2016.

has to be ascribed with some property regarding its roughness, it would be misleading to consider it as the hole-lining marked in orange in Fig. 5.5b; this is indeed an internal part of the object, which cannot be accessed by a machining tool and cannot therefore undergo a change in its roughness. Similarly, the case depicted in Fig. 5.4a likely reflects with more preciseness than Fig. 5.5a experts' understandings of protrusions.

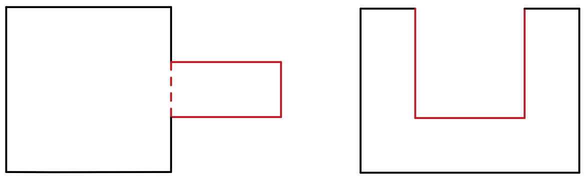

(a) Object with protrused-part (b) Object with holed-part

Figure 5.4: Form feature: second view

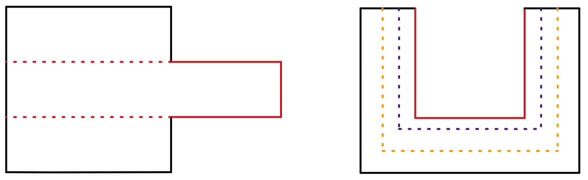

(a) Object with protrused-part (b) Object with holed-part

Figure 5.5: Which form feature?

Finally, in the **third view** a form feature models an immaterial object, i.e., a void space in the sense introduced in [Sect. 5.3]. Differently from the two approaches above, this view is adopted only for hole-like subtractive features, which are also called *negative spaces* (or *negative clearances*)[13] [SLF+13, SG05] or, more explicitly, *voids* [Int16, Ros90]. This perspective on subtractive form features is particularly adopted for assembly modelling, when experts need to refer to void spaces in material objects where some component can be allocated [RS15]. Clearly, in order for this perspective to represent the broad spectrum of form features and not only the subtractive ones, it needs to be integrated with one of the approaches previously mentioned. Fig. 5.6 shows an immaterial hole; differently from the previous views, here the hole is the void space marked in grey.

[13]This terminology is due to the fact that, as previously said, hole-like form features are obtained by material removal; a subtractive feature is thus 'negative' in the sense of lacking material.

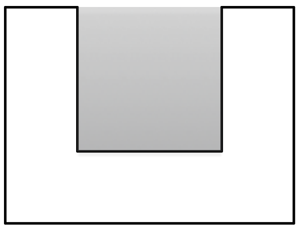


Figure 5.6: Form feature: third view; object with immaterial hole

Discussion. When we look at the three views above, experts shift from one to the other depending on application contexts and background conceptualisations. For instance, an immaterial understanding of subtractive features may be better suited for assembly modelling, whereas a material view may be required if experts want to talk about (what they think as) the material surfaces of holes. In some case the three readings are mixed within the same information modelling resource at the expenses of consistency [Anj11]. From an ontological perspective, indeed, the classification of an individual entity as both a shape quality, a material and an immaterial object leads into inconsistency, because of the incompatible properties of these entities. A hole as an immaterial object (third view), indeed, is not constituted by material; differently, a hole is a material object if understood as a product's part with a characterising shape (second view). Finally, it would be misleading to talk about the materiality of qualities, since these are not entities that can be constituted by materials.

From our perspective, on the one hand, the three views may be separately kept, as long as they are clearly represented in an ontological coherent manner that explicitly acknowledges for their differences. However, on the other hand, this approach may raise some interoperability issue when, e.g., feature-based models developed according to different perspectives are shared and eventually integrated.

In the next chapter we shall firstly look at how the three modelling views presented through this section can be formally represented in our approach. We then look at some modelling issues that each perspective has to face and investigate how the three views can be integrated within a unified perspective.

5.6 Comparison with the literature

Holes in the ontological literature The understanding of holes as immaterial objects is broadly adopted in ontology engineering. In the development of formal ontologies for bio-medicine, for example, the representation of holes is necessary along with the representation of human anatomy. In [Don04] a hole is a void space in, e.g., a blood vessel within which blood flows. In ontologies for geology, holes are void spaces in minerals that allow for the storage of liquids [HB12]. In these views, a hole is a void space in an object in the sense that it is not constituted by any material, but it does not for this coincides with a region of space. A hole, indeed, moves when its host moves, whereas regions cannot move [Gal00], at least if space is understood in the absolute sense as a "container" where physical objects are located. Donnelly [Don05] suggests to look at holes as subkinds of *relevant places*, which are regions of space that are relative to certain objects and therefore move along with them. In [HB12] a physical hole is a physical void that is located in a void region; the latter is a spatial region within the convex hull of an object (the hole's host) that is disjoint from the object occupied region. Finally, in [CV94, Gal00] holes are immaterial objects that depend on their hosts; the authors however do no represent void regions.

Components. Ontologies and product models maintain a sharp distinction between constructional parts and assemblies [RHF$^+$06, KYDH06]. However, the terminology can vary across and within communities, sometimes because of compliance with standards. For example, Lin and colleagues [LFB96], who presented one of the first ontologies for product knowledge, use 'part' instead of 'product' and distinguish between 'primitive part' and 'assembly'. A similar terminology is employed by [UYC$^+$13, CY11, IY15]. In [RRB15, SFFK$^+$03, AHYC12a, ZS04], among others, components are modelled as engineering features (called *structural features* in [RRB15, ZS04]), although the difference between, e.g., a hole feature and a component feature is not explicitly addressed.

Form features. At the state of art, form features are formally represented in different ways depending on application scenarios.

Traditional approaches in engineering adopt mathematical formalisms, which employ geometric notions like line, vertex, concavity, perpendicularity [ZQH$^+$07, PH96, FGM99]. These approaches are applied in a variety of ways, e.g., to support recognition algorithms to individuate features in CAx models. The core application of these works is not indeed the qualitative representation of form features but their quantitative modelling in terms of geometric constraints. Additionally, these approaches do not specify what form features are, or how they distinguish from non-feature elements. Even approaches that are meant to provide the basic, high-level modelling constructs for feature-based representations usually model form features as the aggregation of low-level geometric entities without explicitly saying to which domain entities a form feature refers [RRB15, TCM13]. Since the ontological nature of form feature is not taken into account, there is no surprise that (semantically) underspecified relationships between features and products are stated.

In another modelling approach, engineering features are represented in logical formalisms, even though the semantics of geometric and topological notions is commonly left unspecified. In [CY11], for example, a cylindrical hole (form feature) is represented as an entity with connected circular surfaces, even though connection, circularity and surfaces are not formally characterised. In [WY14, WPY10, EKK$^+$13] the authors combine OWL taxonomies with SWRL rules. Also in these cases, however, the meaning of geometric notions is not formally captured. Differently, Borgo and colleagues [BGM96] propose a mereo-geometry[14] and show how it can be applied for feature-based modelling. More recently, the application of mereotopologies to formalise assembly constraints obtained by means of engineering features has been investigated in [KYK08, DMK12, GDD$^+$15]. However, the application of the proposed theories is threaten by formal flaws. Moreover, the approaches in [KYK08, DMK12, GDD$^+$15] lack an ontological theory by which to make sense of fundamental distinctions between the represented entities.

Other proposal abstract from geometric modelling and focus on the qualitative representation of form features for data sharing and interoperability across CAx systems. The overall idea is to abstract from geometry and to avoid reducing form features to the aggregation of lines, points or surfaces [ElM91, DPD16]. From this perspective, a form feature conveys qualitative design intents, or quantitative properties which are not however geometrically expressed. For example, in [FFBS08, RHF$^+$06, UYC$^+$13, CCF$^+$09], among others, holes and the like are characterised by shape qualities, which are not geometrically represented.

Our approach contextualises within these latter approach, since we do not aim at providing a geometry-like formalism, but rather an ontological coherent way for feature-based product knowledge representation by which design intents can be conveyed.

[14]Mereo-geometries are logical theories that combine mereological, topological and geometrical primitives [BM10].

Chapter 6

Formal representation

In this chapter we present the formal representation of some of the notions informally introduced in the previous two chapters. We focus only on the notions by which a high-level ontological characterisation of engineering features can be provided. In [Sect. 6.2] ontological features are represented in a way that reflects engineering knowledge. [Sect. 6.2] presents the overall approach to distinguish between the type and the physical levels of product knowledge representation. This approach is specialised in [Sect. 6.3] and [Sect. 6.4] to cover the notions of technical product, technical component and engineering feature. Finally, the representation of form features is given in [Sect. 6.5].

Remarks on notation:

- $c_1, c_2, \ldots, c_3$: variables for concepts (CN);
- $r_1, r_2, \ldots, r_3$: variables for regions in quality spaces (SP);
- $x, y, \ldots, z$: variables for physical entities (OB, E); also, variables within mixed predicates, e.g., (Def13);
- $t_1, t_2, \ldots, t_3$: variables for temporal qualities (TQ);
- $q_1, q_2, \ldots, q_3$: variables for qualities (Q) other than temporal ones;
- Uppercase acronyms: predicates taken from previous works, e.g., `CF` for DOLCE-CORE relationship of *classification*;
- Lowercase bold labels: relational predicates hereby introduced, e.g., **`hosts`** for the relationship of *hosting*;
- Lowercase italic labels: constants for regions in quality spaces and concepts; e.g., $fiat500$ (the concept of *being a fiat500*); *red* (the red region within the colour quality space);
- Lowercase italic label#: constant for a physical particular; e.g., $my\_car\#$ (the constant referring to my physical car).

6.1 Ontological features

As we saw in [Sect. 5.3] ontological features are objects that, whenever are present, are related to other objects, namely their *hosts*, and cannot exist if detached from them. In order to characterise the link between an ontological feature and the corresponding host, we firstly introduce a general relationship of *hosting* between objects.

By axiom (Ax4), *hosting* holds between either material or immaterial objects; $\texttt{hosts}(x, y)$ is read as "x hosts y". The axiom establishes that at all times in which the hosted object y is present, x is present, too. For example, if a car's bodywork has a bump, this is represented by saying that the bodywork hosts the bump, whose presence implies the presence of the bodywork. By (Ax5) and (Ax6) *hosts* is irreflexive and asymmetric.

Note that (Ax4) establishes the existential dependence of y on a *specific* x; this means that y does not survive a host replacement and if x stops existing, y stops existing, too. In the philosophical literature, Casati and Varzi [CV94] assume that when a hole is the entity being hosted, it just generically depends on its host; this means that the host can change without affecting the hole identity. Our perspective, although based on philosophical theories, is driven by engineering conceptualisations. From an engineering perspective, it is indeed reasonable to assume that hole-like (or bump-like) entities are related to specific objects and therefore cannot undergo hosts replacement, nor they can continue existing when the identity of their hosts changes. We thus assume that an object host is unique (Ax7), namely, if y is hosted in both x and x', than x and x' are the same.[1]

Ax4 $\texttt{hosts}(x, y) \rightarrow (\textsc{ImmaterialObject}(x) \vee \textsc{MaterialObject}(x)) \wedge (\textsc{ImmaterialObject}(y) \vee \textsc{MaterialObject}(y)) \wedge \forall t\ (\texttt{PRE}(y, t) \rightarrow \texttt{PRE}(x, t))$
(*host* holds between either material or immaterial objects; also, if x hosts y, whenever y is present, x is present, too.)

Ax5 $\neg\ \texttt{hosts}(x, x)$
(*hosts* is irreflexive)

Ax6 $\texttt{hosts}(x, y) \rightarrow \neg\ \texttt{hosts}(y, x)$
(*hosts* is asymmetric)

Ax7 $\texttt{hosts}(x, y) \wedge \texttt{hosts}(x', y) \rightarrow x = x'$
(if y is hosted in both x and x', then x and x' are the same entity)

By axiom (Ax8), we introduce the primitive relationship of *indirect hosting*, written as $\texttt{indHosts}(x, y)$. Accordingly, x indirectly hosts y if an only if x hosts y, or there exists a z such that x indirectly hosts z and z hosts y.

[1]When a hole is drilled through two assembled objects, e.g., the components making up an assembled product, the hole is hosted in the whole assembly.

For example, the axiom allows us to say that if a hole hosts its own (immaterial) surface and the hole is hosted in a gear, the gear indirectly hosts the hole surface. A similar constraint can be introduced to talk about indirect hosting when the host is part of another object. For instance, we may want to say that if a bodywork hosts a bump and the bodywork is part of a car, then the car indirectly hosts the bump. By (Def2), if z hosts y and z is a proper part of x at time t, this means that x indirectly hosts y at t according to *parthood* ($\texttt{indHosts}_{PP}(x, y, t)$). The relationship is temporally qualified and the t argument refers to the time at which *parthood* holds between z and x.

Ax8 $\texttt{indHosts}(x, y) \leftrightarrow \texttt{hosts}(x, y) \vee \exists z(\texttt{indHosts}(x, z) \wedge \texttt{hosts}(z, y))$
(x indirectly hosts y if an only if x hosts y or there exists a z such that x indirectly hosts z and z hosts y)

Def2 $\texttt{indHosts}_{PP}(x, y, t) \triangleq \exists z(\texttt{PP}(z, x, t) \wedge \texttt{hosts}(z, y))$
(x indirectly hosts y at t according to *parthood* means that there is a proper part z of x, such that z hosts y)

A hosted object can in turn host something else; as we saw in one of the previous examples, a hole that is hosted in a gear hosts its own (immaterial) surface. By (Ax9) we establish that if x hosts y, either x is not hosted, or it is indirectly hosted in a v, which in turn is not hosted. This guarantees that whenever an object is hosted, the hosting chain has to come to an end and cannot infinitely regress. (Note however that we do not know a priori how long a hosting chain can be.)

Ax9 $\texttt{hosts}(x, y) \rightarrow \neg\exists z(\texttt{hosts}(z, x)) \vee$
$\exists v \forall w(\texttt{indHosts}(v, x) \wedge \neg\texttt{hosts}(w, v))$
(if x hosts y, then either x is not hosted, or it is indirectly hosted in a v, which in turn is not hosted)

Axioms (Ax10)–(Ax13) constrain the relations between *hosting* and *parthood* between material and immaterial objects. By (Ax10), if x is part of y at t and y is a material object, then x is a material object. Similarly, if x is part of y and y is an immaterial object, then x is an immaterial object, see axiom (Ax11). By (Ax12), if a material object is hosted in another material object, then they stand in the relationship of *constant parthood*.[2] By (Ax13), any object y hosted in an immaterial object x is an immaterial constant part of x.

Ax10 $\texttt{P}(x, y, t) \wedge \textsc{MaterialObject}(y) \rightarrow \textsc{MaterialObject}(x)$
(if x is part of y at t and y is a material object, then x is a material

[2]Recall from [Sect. 3.3] that *constant parthood* (CP) is defined in DOLCE-CORE; accordingly, x is a constant part of y means that at all times t in which x is present, x is part of y at t.

object, too)

Ax11 $\mathtt{P}(x, y, t) \wedge \textsc{ImmaterialObject}(y) \rightarrow \textsc{ImmaterialObject}(x)$
(if x is part of y at t and y is an immaterial object, then x is an immaterial object, too)

Ax12 $\textsc{MaterialObject}(x) \wedge \mathtt{hosts}(x, y) \wedge \textsc{MaterialObject}(y) \rightarrow \mathrm{CP}(y, x)$
(if a material object y is hosted in a material object x, y is a constant part of x)

Ax13 $\textsc{ImmaterialObject}(x) \wedge \mathtt{hosts}(x, y) \rightarrow \textsc{ImmaterialObject}(y) \wedge \mathrm{CP}(y, x)$
(any object y hosted in an immaterial object x is an immaterial constant part of x)

By means of these axioms, we define the notion of *ontological material feature* (*material feature* for short) as a hosted material object (Def3), and *ontological immaterial feature* (*immaterial feature*) as a hosted immaterial object (Def4). By (Def5), an ontological feature (F) is either a material or an immaterial feature.[3]

Def3 $\textsc{MaterialFeature}(x) \triangleq \textsc{MaterialObject}(x) \wedge \exists y\ (\mathtt{hosts}(y, x))$
(a material ontological feature x is a hosted material object)

Def4 $\textsc{ImmaterialFeature}(x) \triangleq \textsc{ImmaterialObject}(x) \wedge \exists y\ (\mathtt{hosts}(y, x))$
(an immaterial ontological feature x is a hosted immaterial object)

Def5 $\mathrm{F}(x) \triangleq \textsc{MaterialFeature}(x) \vee \textsc{ImmaterialFeature}(x)$
(an ontological feature is either a material or an immaterial feature)

An example of material feature is a bump, whereas a hole is an example of immaterial feature.[4] These are ontological features because cannot be present in space-time if detached from other objects.

[3]We use the acronym F for ‘ontological feature’ in order to avoid confusion with the engineering notion.

[4]For the sake of the example, we assume holes as immaterial objects, but as stated earlier in [Sect. 5.3] this is not necessarily the case. See [Sect. 6.5] for a further discussion about the ontological nature of holes in engineering.

6.2 Design properties

In [Ch. 4] we informally introduce design properties as properties that are developed and used within engineering contexts for product modelling. Recall that it is relevant to distinguish between physical objects, e.g., technical products and their components, and the design properties that these objects are meant to satisfy, properties that are accurately defined during designing tasks and represented in product specifications (see Fig. 4.4 in [Sect. 4.2]). From now on, we shall talk about the *physical* and the *type* (or design) levels of product knowledge representation to refer to physical objects and design properties, respectively. As we previously saw, the distinction between the two levels is common in standards like PSL and IFC (see [Sect. 4.2]).

Among design properties, we distinguish those that are created (and can be eventually destroyed) from those that are assumed in the experts' backgrounds within larger conceptual systems. For instance, the property of *being a gear* is created at a certain time in terms of *shape*, *dimension*, *colour* and *weight*, among others, where each property is associated to a certain value (e.g., 5kg weight, cylindrical shape). Also, one may assume that these latter properties are not created *ex novo*, but are already given in experts' backgrounds. We do not list the design properties that are created and the ones that are assumed, since this choice may depend on specific modelling applications. Colours, for example, may be either given for granted in experts' conceptual systems, or considered as being created in designing activities. The distinction between created and assumed design properties is more motivated by the designing practise rather than by an ontological distinction between design properties. In designing tasks, indeed, on the one hand experts assume a variety of notions, which they have learned during their studies and practises. On the other hand, by means of these notions, they elaborate and create new notions.

In DOLCE-CORE properties are represented either by extensional predicates, or by concepts or by quality spaces. Concepts are used for properties that do not reduce to their extensions and are related to some contextual aspect, e.g., they depend on some agent. For example, one may distinguish between the properties *being a Bosch-hammer* and *being a Demoltech-hammer*, because they are developed by different companies, even though both properties may classify exactly the same particulars. Predicates are adequate for the basic elements of an ontology, namely to capture the entities that are assumed to exist in the domain of quantification and do not depend on contexts. Thus, differently from concepts, predicates reduce to their extensions. Finally, quality spaces organise individual qualities of the same (quality) kind, whereas the structure of a space depends on some reference conceptual system, e.g., the values provided by a measurement instrument.

Accordingly, we adopt concepts (CN) to model design properties for which we want to stress their being the results of designing development tasks and quality spaces (SP) for design properties that are assumed within experts' conceptual systems. For instance, *being a gear* (*gear* for short) is a concept, whereas *weight* and *shape* are quality spaces whose structure is defined according to some background knowledge. We might also use extensional predicates for some design properties; e.g., *gear* may be taken as a basic element in design conceptualisations. However, we treat it as a DOLCE-CORE concept, by stressing in this way that it is a property intentionally created to meet some engineering requirement.

Axiom (Ax14) introduces the primitive notion of *design concept*, which informally captures design properties that are created in designing tasks.

Ax14 $\textsc{DesignConcept}(c) \rightarrow \mathrm{CN}(c)$
(a design concept c is a DOLCE-CORE concept)

We now introduce relationships between design concepts in order to capture some useful constraints between their instances for product modelling purposes. In particular, we present two relationships, *qualitative characterisation* ($\mathtt{CH}_Q$) and *structural characterisation* ($\mathtt{CH}_{ST}$).

Qualitative characterisation (**Q-characterisation** for short) holds between design concepts and regions in quality spaces. The definition (Def6) says that when a design concept c is Q-characterised by r_1, the latter is the smallest region within the space to which r_1 belongs. (Recall from [Sect. 3.5] that *classification* ($\mathtt{CF}$) is the relationship used in DOLCE-CORE to state that an entity at a certain time is an instance of a concept.)

Def6 $\mathtt{CH}_Q(c, r_1) \triangleq \textsc{DesignConcept}(c) \wedge \forall xt\ (\mathtt{CF}(c, x, t) \rightarrow \exists q r_2\ (\mathtt{I}(q, x) \wedge \mathtt{L}(r_2, q, t) \wedge \mathtt{P}(r_2, r_1))) \wedge \forall r_3 \exists r_2 x q t\ (\mathtt{PP}(r_3, r_1) \rightarrow \mathtt{CF}(c, x, t) \wedge \mathtt{I}(q, x) \wedge \mathtt{L}(r_2, q, t) \wedge \neg \mathtt{P}(r_2, r_3))$
(c is **qualitatively characterised** by r_1 means that c is a design concept and whenever it classifies an entity x, there exists a quality q that inheres in x, and q is located in r_1)

Q-characterisation is used to specify that the instances of a design concept are characterised by a quality, which is associated to a specific value. For example, a company may design Italian moka-pots whose weight is 700 grams, among other qualifying properties. Accordingly, *moka_pot* is a design concept Q-characterised by $700gr$, which is a region within the weight quality space (SP_{wg}), see (f7).[5] In this sense, $700gr$ is a design property but, differently from *moka_pot*, it is not created; it rather belongs to designers' shared knowledge and is (re-)used for product development purposes.

[5]Differently from DOLCE-CORE, we assume that a quality kind is associated only to one quality space, which is the space selected by design experts.

f7 $\mathtt{CH}_Q(moka\_pot, 700gr) \wedge \mathrm{SP}_{wg}(700gr)$
(example: the design concept *moka_pot* is Q-characterised by $700gr$, which is a region in the weight space)

The relationship of *structural characterisation* (**S-characterisation**) is defined in terms of other relationships, which are used to specify *hosting* ($\mathtt{hosts}_{CN}$), *parthood* ($\mathtt{P}_{CN}$) and *constitution* ($\mathtt{K}_{CN}$) relationships between concepts and their instances.

By (Def7), we define parthood between design concepts; $\mathtt{P}_{CN}(c_1, c_2)$ – read as "c_1 is part of c_2" – means that whenever x instantiates c_2, there exists some instance y of c_1 that is part of x. From a high-level perspective, this relationship is useful to represent assembly constraints between products.[6] For example, we can establish that instances of *car* have bodyworks as parts, see (f8).

Def7 $\mathtt{P}_{CN}(c_1, c_2) \triangleq \textsc{DesignConcept}(c_1) \wedge \textsc{DesignConcept}(c_2) \wedge \forall xt(\mathtt{CF}(c_2, x, t) \to \exists y(\mathtt{CF}(c_1, y, t) \wedge \mathtt{P}(y, x, t)))$
(c_1 **is part of** c_2 means that whenever x instantiates c_2, there exists some instance y of c_1 that is part of x)

f8 $\mathtt{P}_{CN}(bodywork, car)$
(example: *bodywork* is part of *car* means that whenever *car* instantiates, there exists an instance of *bodywork* that is part of the instance of *car*)

By (Def8), $\mathtt{hosts}_{CN}(c_1, c_2)$ – read as "c_1 hosts c_2" – means that whenever c_2 instantiates, there exists an instance x of c_1 that hosts y. For example, (f9) says that when instances of $hole_1$ are present, they are related via *hosting* to instances of *gear*.

Def8 $\mathtt{hosts}_{CN}(c_1, c_2) \triangleq \textsc{DesignConcept}(c_1) \wedge \textsc{DesignConcept}(c_2) \wedge \forall yt(\mathtt{CF}(c_2, y, t) \to \exists x(\mathtt{CF}(c_1, x, t) \wedge \mathtt{hosts}(x, y)))$
(c_1 **hosts** c_2 means that whenever y instantiates c_2, there exists an instance x of c_1 that hosts y)

f9 $\mathtt{hosts}_{CN}(gear, hole_1)$
(example: whenever $hole_1$ instantiates, there exists some instance of *gear* that hosts the instance of $hole_1$)

[6] As said in the previous chapter, *parthood* is commonly treated as the most general relationship for assembly representation.

By (Def9), the design concepts c_1 and c_2 are related via *constitution*; $\mathtt{K}_{CN}(c_1, c_2)$ – read as "c_1 constitutes c_2" – means that whenever x instantiates c_2, there exists some y that instantiates c_1, such that y constitutes x. For example, instances of *gear* are made of metal, see (f10).

Def9 $\mathtt{K}_{CN}(c_1, c_2) \triangleq \textsc{DesignConcept}(c_1) \wedge \textsc{DesignConcept}(c_2) \wedge$
$\forall xt\ (\mathtt{CF}(c_2, x, t) \rightarrow \exists y\ (\mathtt{CF}(c_1, y, t) \wedge \mathtt{K}(y, x, t)))$
(c_1 **constitutes** c_2 means that whenever x instantiates c_2, there exists some y that instantiates c_1, such that y constitutes x)

f10 $\mathtt{K}_{CN}(metal_1, gear)$
(example: instances of *gear* are constituted by instances of $metal_1$)

S-characterisation – read as "c_1 is structurally characterised by c_2" – is defined in (Def10) as either *parthood*, *hosting* or *constitution* between design concepts. The relationship is useful from a high-level to represent structural aspects of products, namely to talk about their components, ontological features and constituting materials (see the comment below).

Def10 $\mathtt{CH}_{ST}(c_1, c_2) \triangleq \mathtt{P}_{CN}(c_2, c_1) \vee \mathtt{K}_{CN}(c_2, c_1) \vee \mathtt{hosts}_{CN}(c_1, c_2)$
(c_1 **is structurally characterised by** c_2 means either that c_1 has c_2 as part, or that c_1 is constituted by c_2, or that c_1 hosts c_2)

We include only structural- and qualitative characterisation as relationships between design concepts, because they capture the constraints we need when discussing about engineering features (see [Sect. 6.6]). Further *characterisation*-like relationships can be introduced, if needed by applications.[7] Also, for each relationship additional constraints may be defined when dealing with specific modelling cases. For example, in order to deal with assembly representations, *parthood* may be strengthen in order to capture specific engineering knowledge.

Finally, by means of the relationships introduced in this section, specific notions of design concepts may be defined. For example, it may be argued that most (if not all) design concepts used in engineering are qualitatively characterised. Then, (something like) a *qualitative design concept* can be easily defined by means of $\mathtt{CH}_Q$. The same for $\mathtt{CH}_{ST}$ for design concepts structurally characterised.

[7]In the next chapter, for the sake of the test cases we introduce further *characterisation*-like relationships to talk about topological connection and functionalities.

6.3 Product types and technical products

In [Sect. 4.2] we present the notion of product type as a design property to be satisfied (at some degree) by an intentionally created object in order for the latter to be a technical product. As said, a product type comprises different design properties, which are related to each other; also, a product type satisfies a unity criterion, whose analysis is not hereby considered.

By axiom (Ax15), **product type** is introduced as a primitive. Accordingly, it is a design concept that classifies only artefacts. Recall that the notion of artefact is heterogeneously conceived across the philosophical and engineering literature (see [Sect. 4.1]). We use it to refer to entities that are not naturally given in the quantification domain and whose existence presupposes an intentional creation. We do not however constrain its meaning axiomatically, so that each user of the ontology can characterise it according to his/her own application requirement. Note that axiom (Ax15) also establishes that the entity classified by a product type is not an ontological feature (F). The idea is that product types are only about (artefactual) entities that can exist on their own, whereas – as we have seen – the existence of ontological features always depends on non-feature objects. This seems a reasonable modelling choice in the domain of product design hereby considered, where products can host ontological features but are not ontological features on their own.[8]

Ax15 $\textsc{ProductType}(c) \rightarrow \textsc{DesignConcept}(c) \wedge \forall xt\ (\texttt{CF}(c,x,t) \rightarrow \textsc{Artefact}(x) \wedge \neg \mathrm{F}(x))$

(if c is a product type, then it is a design concept that classifies artefacts but not ontological features)

By (Ax16), if x is a technical product, x is a material artefact that, whenever is present, is classified by a product type. The axiom also establishes that a product is something that is not hosted. Given the generality of the notion of artefact, we explicitly say in the formula that a product is a material object.

Ax16 $\textsc{TechProduct}(x) \rightarrow \textsc{Artefact}(x) \wedge \textsc{MaterialObject}(x) \wedge \forall t\ (\texttt{PRE}(x,t) \rightarrow \exists c(\textsc{ProductType}(c) \wedge \texttt{CF}(c,x,t))) \wedge \neg\exists y\ (\texttt{hosts}(y,x))$

(if x is a technical product, x is a material artefact that, whenever is present, is classified by a product type c and is not hosted by any y)

The DOLCE-CORE relationship of *classification* holding between product and product type is meant to capture the notion of compliance discussed

[8]In some engineering contexts products are indeed ontological features, e.g., water wells that are realised by removing plots of land without any additional artefactual object such as, e.g., supporting walls. Our notion of product type does not cover this case.

in [Sect. 4.1]. From this perspective, a product x classified by a product type c complies with the properties in c. More specific classification-like relationships can be introduced to explicitly distinguish between partial and total compliance, as discussed in [Sect. 4.1].

A **technical component** is a product explicitly designed to be part of another product, even though it can exist without being part of any (see [Sect. 5.4]). The formal representation of this notion is challenging, because it requires to acknowledge the fact that a component can be *possibly* part of another product. From this perspective, to talk about a component, one has to consider a *possible product* of which the component is part of, an approach requiring an ontological account of possible (artefactual) objects. Following [PBCV11], this may be done by adopting the notion of *engineering possible object*, which is meant to capture an object that only possibly exists in the domain of quantification according to some engineering conceptualisation. For the time being, we rely only on a weak representation of technical components; by (Ax17), if x is a technical component, then x is a technical product.

Ax17 $\textsc{TechComponent}(x) \rightarrow \textsc{TechProduct}(x)$
(if x is a technical component, then x is a technical product)

6.4 Engineering features

We now introduce the formal representation of engineering features, firstly at the type level and then at the physical level, according to the distinction presented in [Sect. 5.2]. The former are called **feature types**, the latter **physical features**.

Feature types. We distinguish between two notions of feature types, namely **structural feature type** and **qualitative feature type**. The former are design concepts that represent structural properties of product types; the latter are used to model qualitative properties (and their values) of both product types and structural feature types. Both notions are formally specified by binary predicates to stress their relational nature. Recall, indeed, that engineering features are always defined in relation to other entities.

By (Def11), c_1 is a structural feature type in c_2 means that c_1 is associated via *S-characterisation* to the product type c_2.[9] By (Def12), a qualitative feature type is the quality space r that Q-characterises either a product type or a structural feature type c_1. (See example below.)

Def11 $\textsc{FeatureType}_{ST}(c_1, c_2) \triangleq \mathtt{CH}_{ST}(c_2, c_1) \wedge \textsc{ProductType}(c_2)$
(c_1 is a **structural feature type** in c_2 means that c_1 structurally characterises c_2, which is a product type)

Def12 $\textsc{FeatureType}_{Q}(r, c_1) \triangleq \mathtt{CH}_{Q}(c_1, r) \wedge (\textsc{ProductType}(c_1) \vee (\exists c_2(\textsc{FeatureType}_{ST}(c_1, c_2))))$
(r is a **qualitative feature type** in c_1 means that r Q-characterises either a product type or a structural feature type c_1)

Recalling the example in (f9), we can represent $hole_1$ as a structural feature type in the product type *gear*, such that $hole_1$ is Q-characterised by the qualitative feature types *cylindrical_shape* and $12mm - diameter$, see (f11) – (f13). To be more precise on the relationship between $hole_1$ and *gear*, we assume that instances of the former are hosted in instances of the latter, see (f11).[10]

f11 $\textsc{FeatureType}_{ST}(hole_1, gear) \wedge \mathtt{hosts}_{CN}(gear, hole_1)$
(example: $hole_1$ is a structural feature type hosted in the product type *gear*)

[9]If needed, one may also add structural sub-features, namely structural features of other structural features, e.g., the immaterial surface of a hole. This is done in the test case presented in [Sect. 7.2].

[10]For the sake of the example, we attribute the dimension qualitative feature type directly to the hole in (f13). We shall better see in [Sect. 6.5] that this representational approach should not be taken for granted.

f12 $\textsc{FeatureType}_Q(\textit{cylindrical\_shape}, \textit{hole}_1)$

(example: *cylindrical_shape* is a qualitative feature type in $\textit{hole}_1$)

f13 $\textsc{FeatureType}_Q(12\textit{mm\_diameter}, \textit{hole}_1)$

(example: $12\textit{mm\_diameter}$ is a qualitative feature type in $\textit{hole}_1$)

By (Def13), **engineering feature type** is defined in terms of either qualitative or structural feature types.

Def13 $\textsc{FeatureType}(x, y) \triangleq \textsc{FeatureType}_Q(x, y) \vee \textsc{FeatureType}_{ST}(x, y)$

(x is an **engineering feature type** in y means that x is either a qualitative or a structural feature type in y)

Physical feature. At the physical level, we distinguish between *qualitative physical feature* (**qualitative ph-feature**) and *structural physical feature* (**structural ph-feature**).

By (Def14), a quality q is a qualitative ph-feature in x means that x is classified by a concept Q-characterised by the qualitative feature type r, so that q is the quality of x located within r.

Def14 $\textsc{PhysicalFeature}_Q(q, x) \triangleq \mathtt{I}(q, x) \wedge \forall t(\mathtt{PRE}(q, t) \rightarrow \exists c r_1 r_2(\textsc{FeatureType}_Q(r_1, c) \wedge \mathtt{CF}(c, x, t) \wedge \mathtt{L}(r_2, q, t) \wedge \mathtt{P}(r_2, r_1)))$

(q is **qualitative ph-feature** in x means that q inheres in x and whenever q is present x is classified by a concept Q-characterised by r_1, so that q is located in the region r_2 within r_1)

By (Def15), x is a structural ph-feature in y means that its classifying concept c_1 is a structural feature type in c_2, which classifies y.

Def15 $\textsc{PhysicalFeature}_{ST}(x, y) \triangleq \forall t(\mathtt{PRE}(x, t) \rightarrow \exists c_1 c_2(\mathtt{CF}(c_1, x, t) \wedge \mathtt{CF}(c_2, y, t) \wedge \textsc{FeatureType}_{ST}(c_1, c_2))$

(x is a **structural ph-feature** in y means that, whenever x is present there are two design concepts, c_1 and c_2, such that c_1 classifying x is a structural feature type in the design concept c_2 that classifies y)

Definition (Def16) defines **engineering physical feature** in terms of either qualitative ph-feature or structural ph-feature.

Def16 $\textsc{PhysicalFeature}(x, y) \triangleq \textsc{PhysicalFeature}_Q(x, y) \vee \textsc{PhysicalFeature}_{ST}(x, y)$

(x is an **engineering physical feature** in y means that x is either a qualitative or a structural physical feature in y)

6.5 Form features: Three modelling views

In this section we represent the three views about form features individuated across the literature and presented in [Sect. 5.5]. The purpose is to show how form features can be represented within our approach according to different conceptualisations. After presenting each view, we discuss their pros and cons by showing the modelling issues they need to face. We then propose how the three views can be integrated in a unified perspective.

In the following subsections, we enumerate axioms by V1, V2 and V3, referring to the first, second and third view, respectively. For example, Def(V1).1 is the first definition in the first view. Also, we represent form features only at the type level; at the physical level the formalisation follows on the lines of (Def14) and (Def15).

6.5.1 First view: Form features for shape qualities

Recall from [Sect. 5.5] that form features are understood as products' shapes according to the first view.

In our approach, this perspective can be represented by defining a form feature as a qualitative feature type r in a design concept c, such that r is a region within the shape quality space (SP_{SH}), see (Def(V1).1). In this sense, all instances of c have (at least) a shape quality.

Def(V1).1 $\textsc{FormFeature}(r, c) \triangleq \textsc{FeatureType}_{Q}(r, c) \wedge \mathrm{SP}_{SH}(r) \wedge \textsc{ProductType}(c)$
(r is a form feature in c means that r is a qualitative feature type, such that r is a region within the shape quality space)

For example, for a gear to have a cylindrical hole means that the gear is cylidrical-hole-like shaped. In formulas, we write that the product type *gear* has the form feature *cylindrical_hole_shape*, see (f(V1.)1).

f(V1.)1 $\textsc{FormFeature}(\mathit{cylindrical\_hole\_shape}, \mathit{gear})$
(example: *cylindrical_hole_shape* is a form feature in *gear*)

A similar modelling approach can be employed for form features like protrusions and bosses. Also in these cases, indeed, for a product to have a protrusion (or boss) means that the product is protrusion-like shaped, i.e., it bears a protrusion-shape quality.

6.5.2 Second view: Form features for shaped parts

In this perspective, a form feature is a physical part of a product characterised by a shape. Despite the explicit reference to products' parts in this view, a form feature is not meant to model components, but – as said throughout the thesis – entities like holes, bosses or protrusions. Again, according to the engineering literature, the core distinction between form features and components is that the former are necessarily related to products and cannot exist if detached from them.

In our approach, by (Def(V2).1) we define a form feature c_1 in c_2 as a structural feature type that is hosted in c_2 and has a qualitative feature type r, which is within the shape quality space. Also, in order to be explicit about the existential dependence between form features and products, we establish that all entities classified by c_1 are material ontological features. Accordingly, all instances of c_1 are hosted in instances of c_2, so that they cannot be present without the latter and bear shape qualities.

Def(V2).1 $\textsc{FormFeature}(c_1, c_2) \triangleq \textsc{FeatureType}_{ST}(c_1, c_2) \wedge \texttt{hosts}_{CN}(c_2, c_1) \wedge \exists r(\textsc{FeatureType}_Q(r, c_1) \wedge \mathrm{SP}_{SH}(r)) \wedge \forall xt(\texttt{CF}(c_1, x, t) \rightarrow \textsc{MaterialFeature}(x))$
(a form feature c_1 in c_2 is a structural feature type hosted in c_2, such that c_1 has a qualitative feature type r for shapes, and classifies only (ontological) material features)

We know from [Sect. 6.2] that material features are parts of their material hosts. Therefore, the entity x classified by a form feature c_1 is part of the product that instantiates c_2.

Recalling the example in the previous section, a gear has a cylindrical hole means that the gear has a material feature as part, which bears a cylindrical shape. In formulas, *hole_part* is the form feature of *gear*, see (f(V2).1); *hole_part* has *cylindrical_shape* as qualitative feature type, see (f(V2).2).

f(V2).1 $\textsc{FormFeature}(\mathit{hole\_part}, \mathit{gear})$
(example: *hole_part* is a form feature in *gear*)

f(V2).2 $\textsc{FeatureType}_Q(\mathit{cylindrical\_shape}, \mathit{hole\_part}) \wedge \mathrm{SP}_{SH}(\mathit{cylindrical\_shape})$
(example: *cylindrical_shape* is a qualitative feature type in *hole_part*)

6.5.3 Third view: Form features for shaped immaterial objects

Recall from [Sect. 5.5] that the third view about form features is only employed for the representation of hole-like subtractive features, which are understood as immaterial objects associated to products.

In our approach, we say that c_1 is an immaterial form feature in c_2 ($\textsc{FormFeature}_{IMT}$), which means that c_1 is a structural feature type hosted in c_2, such that c_1 is characterised by a qualitative feature type r, which is located within a shape quality space; also, c_1 classifies immaterial ontological features, see (Def(V3).1).

Def(V3).1 $\textsc{FormFeature}_{IMT}(c_1, c_2) \triangleq \textsc{FeatureType}_{ST}(c_1, c_2) \wedge \texttt{hosts}_{CN}(c_2, c_1) \wedge \exists r\ (\textsc{FeatureType}_Q(r, c_1) \wedge \mathrm{SP}_{SH}(r)) \wedge \forall xt(\texttt{CF}(c_1, x, t) \rightarrow \textsc{ImmaterialFeature}(x))$

(c_1 is an immaterial form feature in c_2 means that c_1 is a structural feature type hosted in c_2 that has a qualitative feature type r for shapes and that classifies immaterial ontological features)

According to this perspective, to say that a gear has a cylindrical hole means that the it hosts an immaterial form feature. In formulas, $hole_1$ is the immaterial form feature of *gear* and $hole_1$ has the qualitative feature type *cylindrical_shape*, see (f(V3).1) – (f(V3).2).

f(V3).1 $\textsc{FormFeature}_{IMT}(hole_1, gear)$

(example: $hole_1$ is an immaterial form feature in *gear*)

f(V3).2 $\textsc{FeatureType}_Q(cylindrical\_shape, hole_1) \wedge \mathrm{SP}_{SH}(cylindrical\_shape)$

(example: *cylindrical_shape* is a qualitative feature type in $hole_1$ that is located in the quality space of shapes)

6.5.4 Discussion and comparison

In this section we consider three representational needs that often arise in feature-based modelling and look at how each form feature view can deal with them. In [Sect. 6.5.5] we discuss how to integrate the three views within a unified perspective.

Dimensions of form features

The representation of form features is often associated with the representation of form features dimensions. For instance, a hole form feature may be ascribed with both a cylindrical shape and a (nominal) diameter of 12mm with a tolerance value of ± 0.8mm.

From a representational perspective, to enrich a form feature with dimensional information, one may rely on an individual quality that is located in a quality space, which provides specific shapes together with their dimension. In the designing practise, it is indeed common to represent "complex qualities", namely qualities that mix different types of information. For example, according to the first view, when a hole-shape inheres in an object, the latter is characterised by a shape associated to a certain dimension. The same modelling approach can be applied for the second and the third views; in the latter, for example, for an object to have an immaterial hole means to have an hole whose shape is dimensionally characterised.

The assumption behind this approach is that the space for shapes is so structured to provide dimensional information.[11] Also, because dimensions are usually given according to certain measurement systems, the regions of the shape space should be organised with reference to, e.g., the Imperial or the Metric measurement systems.

However, it may be also reasonable to maintain separate the shape and the dimension quality spaces. Gärdenfors [Gär14, Ch.6], for example, reports on approaches for shape representation that do not rely on dimensions, e.g., because they are based on how humans distinguish shapes independently from how big or small they are. Focussing on engineering applications, in representing shapes together with their dimensions, one should choose whether to adopt an approach for the representation of exact quantitative values, or an approach based on a qualitative characterisation of dimensions, or maybe an approach that mixes quantitative and qualitative

[11]For the sake of the discussion, we talk about a quality space for dimension, whereas the definition of such a space may be done in different manners. A quality space, e.g., for diameter qualities may differ from the space for depth qualities; although both qualities are (possibly) measured by the same tools, these are different used in measurement practises, and the quality spaces might thus acknowledge these distinctions. On the other hand, one may have a general space for dimensions and operators that map different qualities (e.g., diameter, depth) to the same space, by specifying how their values are obtained in measurement practises.

values. Some CAx systems used for conceptual design, for example, do not model only the exact quantitative dimension of shapes; they also allow for the qualitative representation of dimensions by using linguistic entries like 'large' or 'small' [DPD16].

The representation of shapes for engineering modelling purposes remains an open issue to be faced in future work. In particular, further research is required to understand how shape and dimension information can be interlinked for CAx applications.

Multiple form features

One and the same product often comprises multiple form features, e.g., two or more holes, or a hole and a protrusion.

According to the first form feature view, one cannot say that one and the same object bears (as a whole) different shapes. However, one may rely on a quality like *being hole-like shaped and being protrusion-like shaped*, which includes both hole-like and protrusion-like qualities. From this perspective, it is therefore needed a quality space that provides complex shapes, e.g., on the basis of shapes used in CAx libraries.

In the second view, when a product has different form features, the product is "cut" across its different material features, each one being characterised by a specific shape. For example, for a table to have both a cylindrical hole and a rectangular protrusion means that the table has two material form features, one with a cylindrical shape and the other one with a rectangular shape.

Finally, in the third view, when the same object has multiple immaterial form features, it hosts different ontological immaterial features, each one with its own shape quality.

Material properties of form features

We have already seen that it is common to distinguish between additive and subtractive form features, both of which can be ascribed with material properties. Recall from [Sect. 5.5] that experts distinguish between different hole-like form features depending on the conditions of (what they think as) their material surfaces (e.g., the surface of a reamed hole is smoother than the one of a drilled hole).

In the first view, a form feature quality cannot be characterised by material properties, since it would be odd to claim that shapes are made of material. It may be, however, reasonable to attribute materiality to the quality's bearer. In this sense, to (loosely) say, e.g., that a protrusion is made of metal actually means that the bearer of the protrusion-shape is made of metal. More precisely, when the bearer is not entirely constituted by the same material, it is necessary to refer to a specific part of the bearer that is made of a certain material. From this perspective in order to represent a product with a protrusion made of metal, one has to distinguish between the protrusion-shape of the product and the part of the product constituted by metal. Then, it would be more correct to say that the protrusion-shape does not inhere in the whole product, but in that part made of metal.

The third view is explicit about the immaterial nature of hole-like form features. Therefore, one cannot talk about the properties of their material surfaces, e.g., their roughness. These properties have to be attributed to the holes' hosts. In the case of, e.g., a drilled hole, one may say that the part of the host surrounding the hole satisfies certain roughness properties.

Only in the second view material properties can be directly ascribed to form features, being these – at the physical level – material features. For example, representing a drilled hole means representing a material feature, whose surface satisfies certain roughness properties.[12]

6.5.5 An integrated perspective

The three form feature views discussed above are three different alternatives to model form features. For example, as said in [Sect. 5.5], the immaterial understanding of hole-like form features may be better suited for assembly design to explicitly model the void space in a product where another product (or some part thereof) can be allocated. If a hole form feature were indeed understood in the material sense, it might be unnatural to claim, e.g., that a shaft has to be inserted *into* a hole, as it is commonly said in engineering [IY15].[13]

From an ontological perspective the assumptions of each view can be coherently represented within a foundational ontology specialised with engineering knowledge. However, as we have seen, each view has to face some issues. To sum up:

- The first view cannot ascribe material properties to form features, e.g., cannot talk about a protrusion made of a specific material, nor about the material surfaces of holes. As said, it would be ontologically misleading to ascribe material properties to (shape) qualities. Also, when a product has multiple form features (e.g., a hole and a protrusion), the shape space has to be structured to allow for the representation of complex shape qualities;

- The second view treats all form features as material entities. Hence, it forces to consider hole-like features in the material sense, whereas the immaterial perspective is useful in some modelling scenarios. Also,

[12]See [Str88, Var15, Smi96, Hah13] for the ontological treatment of surfaces.

[13]Varzi [Var03] claims that the understanding of holes as immaterial objects is of key relevance to make sense of the semantics of spatial containment relationships like 'in'.

some information modelling resources are explicit about the immaterial treatment of hole-like features, e.g., the IFC standard;

- The third view is limited to immaterial form features and therefore, as said, it necessarily needs to be integrated with one of the views above to provide an overall approach for form features representation.

In all three cases, when a form feature is attributed with multiple qualities, e.g., shapes and dimensions, it is necessary to define a quality space that allows to represent their combination.[14]

We propose in the following a unified treatment of form features that reflects the three different conceptualisations discussed above and also provides an overall framework by which form features data can be represented across communities in a similar way. As said in the previous chapter, instead of unifying the three perspectives, one may keep them apart and develop mappings by which data modelled according to a certain view can be compared with data represented according to the other views. The engineering community, however, explicitly calls for integrated modelling approaches by which to make sense of heterogeneous data produced by different communities [EKGT+16, UYC+13]. Integrated approaches, indeed, provide common terminologies and modelling theories that help in overtaking interoperability issues across communities and applications.

We propose to consider form features as structural feature types for the representation of shaped ontological features. Form features, therefore, do not model shape qualities *per se*; they rather represent (dependent) objects that are characterised by shapes, among other qualities. To make sense of the engineering distinction between additive and subtractive form features, we distinguish between material ($\textsc{FormFeature}_{MT}$) and immaterial ($\textsc{FormFeature}_{IMT}$) form features.

Definition (Def17) defines *material form feature*; note that the definition corresponds to (Def(V2).1) as presented in [Sect. 6.5.2]. Accordingly, for a product to have a material form feature, e.g., a bump, means that it hosts an ontological material feature characterised by a shape.

Def17 $\textsc{FormFeature}_{MT}(c_1, c_2) \triangleq \textsc{FeatureType}_{ST}(c_1, c_2) \wedge$
$\texttt{hosts}_{CN}(c_2, c_1) \wedge \exists r(\textsc{FeatureType}_{Q}(r, c_1) \wedge \mathrm{SP}_{SH}(r)) \wedge$
$\forall xt(\texttt{CF}(c_1, x, t) \rightarrow \textsc{MaterialFeature}(x))$
(**material form feature**, as defined in (Def(V2).1))

By (Def18), *immaterial form feature* is defined along with the third view previously presented in [Sect. 6.5.3]. Then, a product has an immaterial form feature means that it hosts a shaped ontological immaterial feature.

[14]The adoption of quality spaces for the representation and organisation of objects qualities is just a modelling approach among others. However, even if a different modelling strategy is adopted, it is anyway relevant to make sense of the fact that the representation of a form feature needs to consider the combination of qualities of different kinds.

Def18 $\textsc{FormFeature}_{IMT}(c_1, c_2) \triangleq \textsc{FeatureType}_{ST}(c_1, c_2) \wedge \texttt{hosts}_{CN}(c_2, c_1) \wedge \exists r\ (\textsc{FeatureType}_{Q}(r, c_1) \wedge \text{SP}_{SH}(r)) \wedge \forall xt(\texttt{CF}(c_1, x, t) \rightarrow \textsc{ImmaterialFeature}(x))$
(**immaterial form feature**, as in defined (Def(V3).1))

Finally, *form feature* is defined in terms of either material or immaterial form features (Def19).

Def19 $\textsc{FormFeature}(c_1, c_2) \triangleq \textsc{FormFeature}_{MT}(c_1, c_2) \vee \textsc{FormFeature}_{IMT}(c_1, c_2)$
(c_1 is a **form feature** in c_2 means that c_1 is either a material or an immaterial form feature in c_2)

Looking back at the three modelling cases discussed above, the integrated approach allows us to deal with them as follows.

(i) *Multiple form features.* This issue can be straightforwardly represented for both material and immaterial form features. As stated earlier, a product has two or more form features in the sense that it hosts different immaterial or material objects. Since form features are not shape qualities *per se*, there is no need to model complex shape qualities, like *being hole-like shaped and being protrusion-like shaped.*[15]

(ii) *Material properties of form features.* The notion of material form feature allows us to explicitly represent form features that are constituted by material. In the case of hole-like form features, although the distinction between material and immaterial form features gives to experts the opportunity to decide whether holes are material or immaterial objects, we propose to represent them in the immaterial sense for the reasons we have discussed throughout the chapter. Then, to (loosely) talk about the material surfaces of holes in our approach means to talk about the surfaces of their hosts. For example, a drilled hole is an immaterial object hosted in a product, such that the product surface (partially) surrounding the hole satisfies some roughness property.

(iii) *Multiple qualities.* As stated earlier, when a form feature bears multiple qualities, the shape space is meant to be complex, e.g., it integrates shape and dimensional values. The definition of complex quality spaces remains however an open issue.

In the test case given in [Sect. 7.1] we show how form features can be represented in our approach.

[15]In the first test case presented in the next chapter [Sect. 7.1], we formally represent a product type with multiple-form features.

6.6 Remarks

As argued throughout the thesis, an ontological understanding of engineering features needs to be provided within a broader ontology for product knowledge representation. For this purpose, we introduce the notion of *design properties*, which are represented in terms of DOLCE-CORE concepts and quality spaces. Accordingly, when emphasis is given to the fact that a property is intentionally created for product modelling needs, it is represented as a concept, which is called *design concept*; the properties that are assumed in experts' conceptual systems are represented as regions in quality spaces. We then introduce the formal representation of *technical product*, *technical component* and *product type*.

Looking at *engineering features*, on the one side they are design properties related to product types; on the other side, they are physical entities related to physical products. This distinction is grasped by the notions of *feature type* and *physical feature*. As we saw, the two notions are strictly related, since the latter are the entities that (generally speaking) satisfy the former.

As said in [Ch. 5], from a high-level perspective engineering features are employed to model ontological features, components, qualities and materials constituting products. In our approach, this can be represented as follows:

- **Engineering features for ontological features.** At the type level they are represented by the notion of *structural feature type* and related to product types via the relationship of *hosting* ($\texttt{hosts}_{CN}$). At the physical level, they are represented as structural physical features. More specifically, we argue that ontological features are the entities represented by form features. Therefore, the more specific notions of *immaterial form feature* and *material form feature* are to be adopted, depending on whether the ontological features at hand are meant to be constituted by material or not;

- **Engineering features for qualities.** At the type level they are represented by the notion of *qualitative feature type*; at the physical level by *qualitative physical feature.* The relationship of *qualitative characterisation* ($\texttt{CH}_Q$) has been introduced to relate qualitative feature types to other design concepts. At the physical level, a quality relates to an object via *inherence*;

- **Engineering features for components.** A component is a technical product that is designed to be (possibly) related via *parthood* to some product. When engineering features model components, structural feature types and structural physical features are to be used at the type and physical levels, respectively. Differently from the case of material ontological features (e.g., a bump), a component does not relate via *hosting* to other products. As said throughout the chapter, indeed, a hosted object cannot exist if detached from its host; e.g., a bodywork bump cannot be detached from the bodywork. Differently, a component does exist even if it is not parthood-related to some other object;

- **Engineering features for constituting materials.** When engineering features are used to model the materials constituting products (and therefore components), they are represented as structural feature types and structural physical features. The relationship used to link them to the constituted objects is *constitution* ($\mathtt{K}_{CN}$ between design concepts; $\mathtt{K}$ between physical entities).

The figure below (Fig. 6.1) shows the taxonomy of the notions represented throughout the chapter. Classes whose diagram is depicted as a dotted box (e.g., *Feature type*) have to be understood as "application classes", i.e., linguistic entries to organise the taxonomy.[16] For example, *Feature type* subsumes both *Structural feature type* and *Qualitative feature type*; the former is subsumed by *Concept*, the latter by *Quality space*. These latter two classes are disjoint in DOLCE-CORE, which means that *Feature type* cannot be subsumed by both *Concept* and *Quality Space*. *Feature type* is however useful to group *Structural feature type* and *Qualitative feature type* under a common upper-level entry.

[16]Application classes are called *domain categories* in [Bor14].

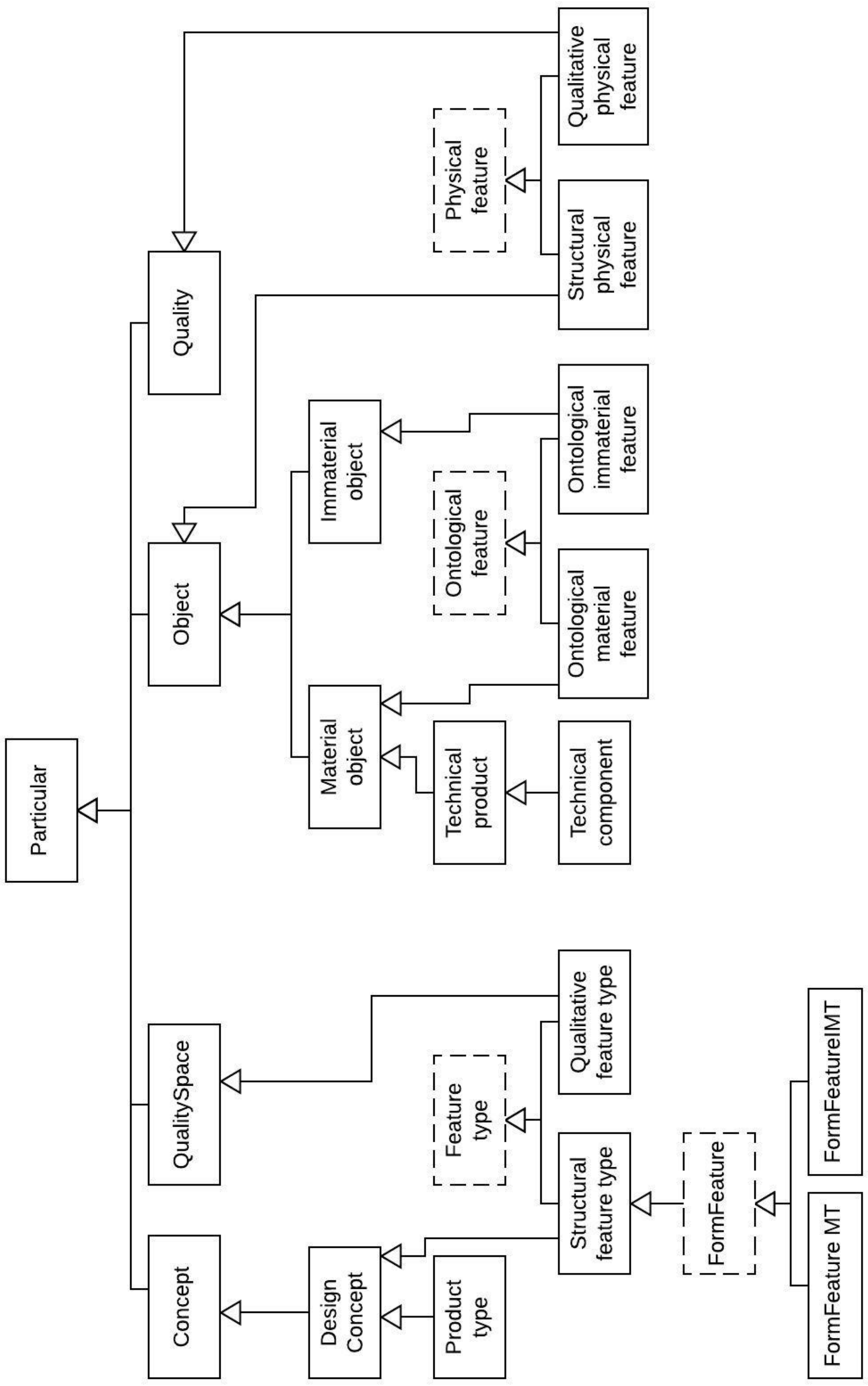


Figure 6.1: Taxonomy

Chapter 7

Case studies

In this chapter we present two cases where the ontology previously presented is applied to represent form features [Sect. 7.1] and to restructure meta-models for feature-based knowledge representation [Sect. 7.2]. In the next chapter we discuss the advantages and limits of our approach.

7.1 Form features representation: An example

In this section we show how the form features of the product in Fig. 7.1, which is called *housing product*, can be represented by means of the ontology given in the previous chapters. (The figure is taken from [Chu10] and slightly modified.)

The form features included in the figure are:

- *ThroughHoleA*, ... *ThroughHoleG*;[1]
- *CounterboreHoleA*;
- *CompoundHoleA*;
- *BossA*;
- *FlangeA*.

Note that *CompoundHoleA* is a hole that comprises both *ThroughHoleG* and *CounterboreHoleA*.

Looking at Fig. 7.1, the first thing to be noticed is that it conveys information at the type level. One could indeed assume that it represents the design properties that instances of the entities depicted in the figure have to satisfy if physically realised. Consider, for example, the *ThroughHoleA*. The hole has a diameter of 12mm with a tolerance of ± 0.8mm. If the hole were

[1] *ThroughHoleC* cannot be seen in the figure.

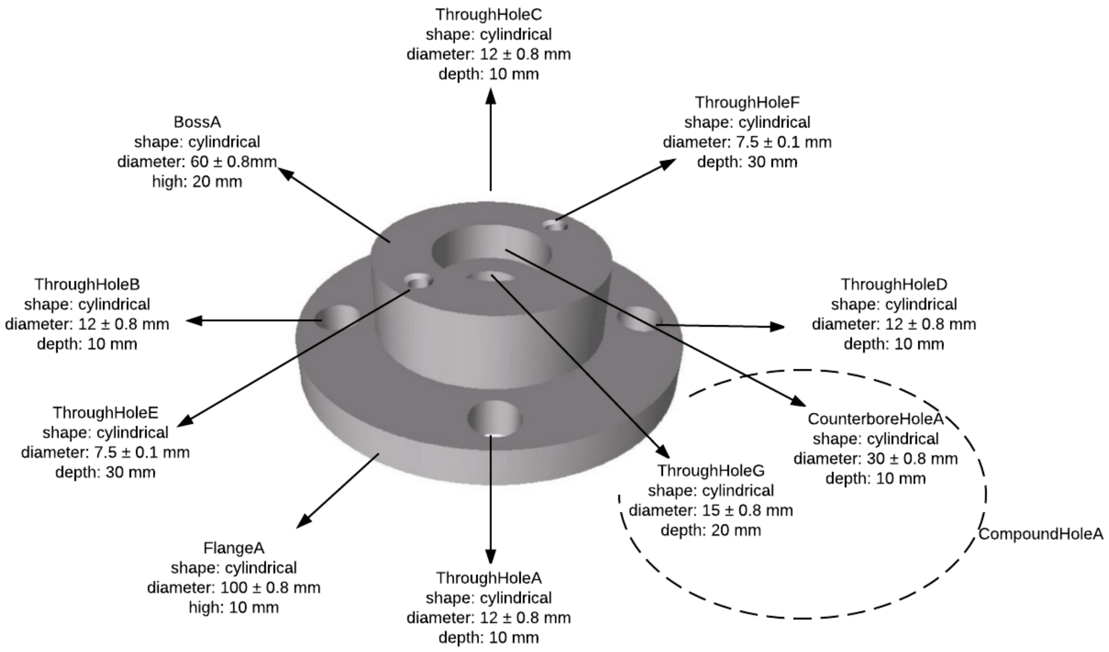


Figure 7.1: Example (adapted from [Chu10])

meant as a physical hole in a product, it would be misleading to attribute it a tolerance value.[2] From this perspective, Fig. 7.1 does not convey any information about physical entities; looking at the figure, we do not know, e.g., whether the diameter of a physical instance of *ThroughHoleA* has a value within the 12±0.8mm tolerance. From the perspective of our ontology, the product depicted in Fig. 7.1 is a product type and each of its engineering features is a feature type. Accordingly, *ThroughHoleA*, ... *ThroughHoleG* and the other features in Fig. 7.1 are (generally speaking) design concepts.

Second, as we have just seen, different features in the figure are similarly named[3] (*ThroughHoleA*, ..., *ThroughHoleG*), although they have different properties. In particular, *ThroughHoleA*, ..., *ThroughHoleD* have the same properties, namely cylindrical shape, depth of 10mm and diameter of 12±0.8mm. This means that when the housing product type is created at the physical level, it will have four *different* physical through holes, among others, whose *different* shape, diameter and depth qualities will (ideally) have the same values. Similarly for *ThroughHoleE* and *ThroughHoleF*, both of which have indeed the same characterising properties. However,

[2]For product knowledge representation purposes, it is reasonable to assume that whenever a physical hole is present, it has a precise diameter.

[3]This labelling choice is due to [Chu10].

their diameter and depth values are not the same of the ones characterising *ThroughHoleA*, ..., *ThroughHoleD*. Finally, the diameter and depth values of *ThroughHoleG* differ from the values of both *ThroughHoleA*, ..., *ThroughHoledD*, and *ThroughHoleE* and *ThroughHoleF*.

From an ontological perspective, it is necessary to understand whether *ThroughHoleA*, ..., *ThroughHoleG* are the same design concept, or they should be rather considered as different ones. This has to do with identity criteria for concepts, an issue that has been extensively discussed in the literature [ML99], although no shared agreement has been reached so far.

For product knowledge representation one may assume that two design concepts are the same when they are characterised by the same properties. Consider, for example, *ThroughHoleA*, ..., *ThroughHoleD*; we know that they are characterised by the same shape, diameter and depth. They may be therefore considered as the same design concept, which differs from the design concept *ThroughHoleE* and *ThroughHoleF*, as well as from the design concept *ThroughHoleG*, given that these are differently characterised.[4]

Following this line of thought, looking at Fig. 7.1, for the sake of the example we distinguish between the following feature types in order to represent the housing product type:

- *hole*: a general form feature for the representation of physical holes;
- *thr_hole* (read as: through hole): a sub-concept of *hole*. It is qualitatively characterised by a (complex) qualitative region, which provides shape information along with its diameter and depth;
- thr_hole_1: a sub-concept of *thr_hole*. It is qualitatively characterised by a qualitative region, which specifies that instances of thr_hole_1 have a cylindrical shape with diameter within the value 7.5±0.1mm and depth of 30mm;
- thr_hole_2: a sub-concept of *thr_hole*. It is qualitatively characterised by a qualitative region, which specifies that instances of thr_hole_2' have a cylindrical shape with diameter within the value 12±0.8mm and depth of 10mm;
- thr_hole_3: a sub-concept of *thr_hole*. It is qualitatively characterised by a qualitative region, which specifies that instances of thr_hole_3 have a cylindrical shape with diameter within the value 15±0.8mm and depth of 20mm;

[4]In this approach it is assumed that the characterising properties of a design concept are always completely specified. However, one may also develop a theory that allows for the partial specification of design concepts to acknowledge their evolution over time. An initial study of concept partial characterisation and evolution for design is presented in [SMP15].

- *ct_hole* (read as: counterbore hole): a sub-concept of *thr_hole* for representing physical counterbore holes;
- *ct_hole*$_1$: a sub-concept of *ct_hole*. It is qualitatively characterised by a qualitative region, by which instances of *ct_hole*$_1$ bear a cylindrical shape with diameter within the value 30±0.8mm and depth of 10mm;
- *cmp_hole* (read as: compound hole): a sub-concept of *hole*; it is used for representing holes that comprise different holes;
- *cmp_hole*$_1$: a form feature that comprises both *thr_hole*_3 and *ct_hole*$_1$;
- *boss*: a general form feature for representing physical bosses. It is qualitatively characterised by a (complex) qualitative region, which provides shape information along with its diameter and height;
- *boss*$_1$: a subconcept of *boss*. It is qualitatively characterised by a qualitative region, which specifies that instances of *boss*$_1$ have a cylindrical shape with diameter within the value 60±0.8mm and height of 20mm;
- $flange$: a form feature for the representation of physical flanges. It is qualitatively characterised by a (complex) qualitative region, which provides shape information along with its diameter and height;
- $flange_1$: a subconcept of $flange$. It is qualitatively characterised by a qualitative region, which specifies that instances of $flange_1$ have a cylindrical shape with diameter within the value 100±0.8mm and height of 10mm.

Some comments are required. First, for the sake of the example, we assume a general *hole feature type*, which specialises in the sub-concepts *thr_hole*, *ct_hole*, *cmp_hole* and their subclasses. This allows us to make sense of the fact that the form features named *ThroughHoleA*, ..., *ThroughHoleG*, *CounterboreHoleA* and *CompoundHoleA* in Fig. 7.1 are holes. The same approach is adopted for *boss* and *flange*, which – as general feature types – are specialised to cover the form features in the example.

Second, *thr_hole*$_1$, *thr_hole*$_2$ and *thr_hole*$_3$ are introduced for representing *ThroughHoleE/ThroughHoleF*, *ThroughHoleA*, ..., *ThroughHoleD* and *ThroughHoleG*, respectively. The introduction of these feature types is necessary to make sense of the idea that some of the holes in Fig. 7.1 share some common property (they are through holes, indeed) while having different dimensions. If we would introduce only one feature type (*through hole*), it would be difficult to say that the product type in Fig. 7.1 comprises different through holes with different dimension properties. The use of specific design concepts is coherent with the overall understanding of concepts presented throughout the thesis; nothing indeed prevents the use of general concepts, which then specialise to classify specific objects.

Third, the feature type for compound holes (*cmp_hole*) is used to represent holes that are designed as being formed from (at least) two other holes. From an ontological perspective, this requires the definition of unity criteria, as well as an approach to represent how different (whole) holes connect to result in a bigger hole. The investigation of unity criteria for holes goes however behind our purposes. The reader can refer to [CV94, Hah13].

Finally, in order to represent form features along with their shapes and dimensions, we assume that the quality space for shapes is complex, so that each shape is associated with information regarding its dimensions. As we saw in the previous chapter, this approach should not be taken for granted and further research is required for a well-established representation of shape spaces for engineering applications.

For the sake of the test case, we introduce the primitive relationship of subsumption ($\sqsubseteq$) between design concepts.[5] The relationship is used to represent taxonomies, from the most general to the most specific design concepts. The formula (tcf.1) establishes that if c_1 is subsumed by c_2, then they are both design concepts and all instances of c_1 are instances of c_2, too.

tcf.1 $c_1 \sqsubseteq c_2 \rightarrow \textsc{DesignConcept}(c_1) \wedge \textsc{DesignConcept}(c_2) \wedge \forall xt\ (\texttt{CF}(c_1, x, t) \rightarrow \texttt{CF}(c_2, x, t))$

(c_1 is subsumed by c_2 if they are both design concepts and all instances of c_1 are instances of c_2, too)

Formulas (tcf.2)–(tcf.11) represent subsumption relationships between the subconcepts of *hole*, *boss* and *flange*.[6]

tcf.2 $thr\_hole \sqsubseteq hole$

(*thr_hole* is a hole)

tcf.3 $thr\_hole_1 \sqsubseteq thr\_hole$

(thr_hole_1 is a through hole)

tcf.4 $thr\_hole_2 \sqsubseteq thr\_hole$

(thr_hole_2 is a through hole)

tcf.5 $thr\_hole_3 \sqsubseteq thr\_hole$

(thr_hole_3 is a through hole)

tcf.6 $cmp\_hole \sqsubseteq thr\_hole$

(*cmp_hole* is a through hole)

tcf.7 $cmp\_hole_1 \sqsubseteq cmp\_hole$

(cmp_hole_1 is a compound hole)

[5]We prefix the formulas below by **tcf.***n* standing for test case formula.

[6]We do not formally define the distinction between, e.g., a through and a counterbore hole. This would require the adoption of a formal language that allows for the representation of topological and geometrical constraints (see, e.g., [Hah13]). As said in [Sect. 5.6], our work addresses the need of a qualitative representation of form features that abstracts from geometry.

tcf.8 $ct\_hole \sqsubseteq thr\_hole$

(*counterbore hole* is a through hole)

tcf.9 $ct\_hole_1 \sqsubseteq ct\_hole$

(ct_hole_1 is a counterbore hole)

tcf.10 $boss_1 \sqsubseteq boss$

($boss_1$ is a boss)

tcf.11 $flange_1 \sqsubseteq flange$

($flange_1$ is a flange)

Formulas (tcf.12)–(tcf.19) say that thr_hole_1, thr_hole_2, thr_hole_3, ct_hole_1, cmp_hole_1, $boss_1$ and $flange_1$ are (material or immaterial) form features in the product type *housing_product*. (Formula (tcf.12) can be actually derived from, e.g., (tcf.13) together with (Def18) and (Def11). It is however introduced for clarity.)

tcf.12 $\textsc{ProductType}(housing\_product)$

(*housing_product* is a product type)

tcf.13 $\textsc{FormFeature}_{IMT}(thr\_hole_1, housing\_product)$

(thr_hole_1 is an immaterial form feature in housing_product)

tcf.14 $\textsc{FormFeature}_{IMT}(thr\_hole_2, housing\_product)$

(thr_hole_2 is an immaterial form feature in housing_product)

tcf.15 $\textsc{FormFeature}_{IMT}(thr\_hole_3, housing\_product)$

(thr_hole_3 is an immaterial form feature in housing_product)

tcf.16 $\textsc{FormFeature}_{IMT}(ct\_hole_1, housing\_product)$

(ct_hole_1 is an immaterial form feature in housing_product)

tcf.17 $\textsc{FormFeature}_{IMT}(cmp\_hole_1, housing\_product)$

(cmp_hole_1 is an immaterial form feature in housing_product)

tcf.18 $\textsc{FormFeature}_{MT}(boss_1, housing\_product)$

($boss_1$ is a material form feature in housing_product)

tcf.19 $\textsc{FormFeature}_{MT}(flange_1, housing\_product)$

($flange_1$ is a material form feature in housing_product)

Formulas (tcf.20)–(tcf.25) represent the qualitative characterisation of the form features introduced above. For example, formula (tcf.20) says that an instance of thr_hole_1 has a quality with cylindric shape (*cy_sh*), which is deep 30mm (*30mmDP*) and whose diameter ranges from 7.4mm to 7.6mm (7.4_7.6$mmDM$). In this sense, *30mmDP;7.4_7.6mmDM;cy_sh* is a non-atomic region in a complex qualitative space, which provides shape information along with its depth and diameter. (As said, the formal representation of regions within complex quality spaces requires further research.)

tcf.20 $\text{FEATURETYPE}_Q(30mmDP; 7.4\_7.6mmDM; cy\_sh, thr\_hole_1)$ (an instance of thr_hole_1 has a quality with cylindric shape (cy_sh), which is deep 30mm ($30mmDP$) and whose diameter ranges from 7.4mm to 7.6mm ($7.4\_7.6mmDM$))

tcf.21 $\text{FEATURETYPE}_Q(10mmDP; 11.2\_12.8mmDM; cy\_sh, thr\_hole_2)$ (an instance of thr_hole_2 has a quality with cylindric shape (cy_sh), which is deep 10mm ($10mmDP$) and whose diameter ranges from 11.2mm to 12.8mm ($11.2\_12.8mmDM$))

tcf.22 $\text{FEATURETYPE}_Q(20mmDP; 14.2\_15.8mmDM; cy\_sh, thr\_hole_3)$ (an instance of thr_hole_3 has a quality with cylindric shape (cy_sh), which is deep 20mm ($20mmDP$) and whose diameter ranges from 14.2mm to 15.8mm ($14.2\_15.8mmDM$))

tcf.23 $\text{FEATURETYPE}_Q(10mmDP; 29.2\_30.8mmDM; cy\_sh, ct\_hole_1)$ (an instance of ct_hole_1 has a quality with cylindric shape (cy_sh), which is deep 10mm ($10mmDP$) and whose diameter ranges from 29.2mm to 30.8mm ($29.2\_30.8mmDM$))

tcf.24 $\text{FEATURETYPE}_Q(20mmHG; 59.2\_60.8mmDM; cy\_sh, boss_1)$ (an instance of $boss_1$ has a quality with cylindric shape (cy_sh), which is high 20mm ($20mmHG$) and whose diameter ranges from 59.2mm to 60.8mm ($59.2\_60.8mmDM$))

tcf.25 $\text{FEATURETYPE}_Q(10mmHG; 99.2\_100.8mmDM; cy\_sh, flange_1)$ (an instance of $flange_1$ has a quality with cylindric shape (cy_sh), which is high 10mm ($10mmHG$) and whose diameter ranges from 99.2mm to 100.bmm ($99.2\_100.8mmDM$))

For the sake of the test case the formulas (tcf.26)–(tcf.27) are introduced to represent that *housing_product* comprises multiple form features with the same properties. By (tcf.26), whenever x instantiates *housing_product*, there are two instances of thr_hole_1 that are hosted in x. Similarly, (tcf.27) says that when x instantiates *housing_product*, there are four instances of thr_hole_2 hosted in x.

tcf.26 $\mathtt{CF}(\mathit{housing\_product}, x, t) \rightarrow \exists yz\ (\mathtt{CF}(\mathit{thr\_hole}_1, y, t) \wedge \mathtt{CF}(\mathit{thr\_hole}_1, z, t) \wedge \mathtt{hosts}(x, y) \wedge \mathtt{hosts}(x, z) \wedge \neg(y = z) \wedge \forall v\ (\mathtt{CF}(\mathit{thr\_hole}_1, v, t) \wedge \mathtt{hosts}(x, v) \rightarrow (v = y \vee v = z)))$
(if x instantiates *housing_product*, there are two instances of *thr_hole*$_1$ that are hosted in x)

tcf.27 $\mathtt{CF}(\mathit{housing\_product}, x, t) \rightarrow \exists yzvw\ (\mathtt{CF}(\mathit{thr\_hole}_2, y, t) \wedge \mathtt{CF}(\mathit{thr\_hole}_2, z, t) \wedge \mathtt{CF}(\mathit{thr\_hole}_2, v, t) \wedge \mathtt{CF}(\mathit{thr\_hole}_2, w, t) \wedge \mathtt{hosts}(x, y) \wedge \mathtt{hosts}(x, z) \wedge \mathtt{hosts}(x, v) \wedge \mathtt{hosts}(x, w) \wedge \neg(y = z) \wedge \neg(y = v) \wedge \neg(y = w) \wedge \neg(z = v) \wedge \neg(z = w) \wedge \neg(v = w) \wedge \forall u\ (\mathtt{CF}(\mathit{thr\_hole}_2, u, t) \wedge \mathtt{hosts}(x, u) \rightarrow (u = y \vee u = z \vee u = v \vee u = w)))$
(if x instantiates *housing_product*, there are four instances of *thr_hole*$_2$ that are hosted in x)

Finally, as we have seen, the form features called $ThroughHoleG$ and $CounterboreHoleA$ in Fig. 7.1 form a bigger hole that comprises both of them. This can be represented by saying that *cmp_hole*$_1$ is related via parthood ($\mathtt{P}_{CN}$) to both *thr_hole*$_3$ and *ct_hole*$_1$. However, to stress that an instance of *cmp_hole*$_1$ is a hole that comprises instances of the two latter form features as *connected* parts, we need to add a further constraint. This may be done by introducing a (topological) connection-like relationship between design concepts, by which the instances of the related concepts are topologically connected. In (tcf.28) we use the primitive predicate C_{CN} to say that instances of *thr_hole*$_3$ and *ct_hole*$_1$ are topologically connected.

tcf.28 $\mathtt{P}_{CN}(\mathit{thr\_hole}_3, \mathit{cmp\_hole}_1) \wedge \mathtt{P}_{CN}(\mathit{ct\_hole}_1, \mathit{cmp\_hole}_1) \wedge C_{CN}(\mathit{thr\_hole\_3}, \mathit{ct\_hole}_1)$
(an instance of *cmp_hole*$_1$ comprises instances of *thr_hole*$_3$ and *ct_hole*$_1$ as connected parts)

7.2 Towards a feature-based meta-model

The engineering community lacks a common way to represent features suitable to support data sharing and interoperability between systems and communities. To tackle this problem some UML meta-models have been proposed. In particular, the works presented by [TCM13, RRB15] go in our direction, since they are both meant to provide the high-level framework to create feature-based models. Nevertheless, these meta-models have not been developed on the grounds of an ontological theory for product knowledge representation and suffer from some flaws when looked from an ontological perspective.

In this section we consider how the ontological understanding of engineering features presented in the previous chapters can provide the ontological basis for the meta-models proposed in [TCM13, RRB15]. From this perspective, our ontology acts as a methodology for improving the semantic transparency of existing works.

The section is structured as follows. We first introduce the meta-models developed in [TCM13, RRB15] and then address their drawbacks from an ontological perspective. Finally, we show how the meta-models can be restructured and integrated by means of our ontology.

Note that the notion of meta-model is used with different meanings across the literature. In some case, given a diagrammatic language like UML, the set of the available graphic modelling primitives forms the so-called *concrete syntax* of the language, whereas the meta-model of the language defines the rules for the creation of well-formed models in that language [Gui05]. In some other case, a meta-model is a model that does not describe a particular situation in the domain at stake, but it is rather employed to provide a general picture of the domain. From this perspective, there is tight relation between ontologies and meta-models [HS12].

7.2.1 Unified Feature Model

The Unified Feature Model is a meta-model aimed at providing a high-level conceptual structure to manage and share feature data [TCM13, CMTT06, CTCM13]. The representation of the meta-model in UML is showed in Fig. 7.2.

Generic Feature (previously called *Unified Feature* [CMTT04]) is "the most basic [...] entity template" [TCM13, p.90], i.e., the key class in the meta-model for the representation of features. Indeed, any feature class (e.g., form or functional feature) is meant to be subsumed by *Generic Feature*. As showed in Fig. 7.2, *Generic Feature* is related to five classes, namely *Attribute*, *Topological Entity*, *Parameter*, *Constraint* and *Feature Model*.

Attribute plays a fundamental role within the meta-model to represent qualitative or quantitative information, including relationships. It spe-

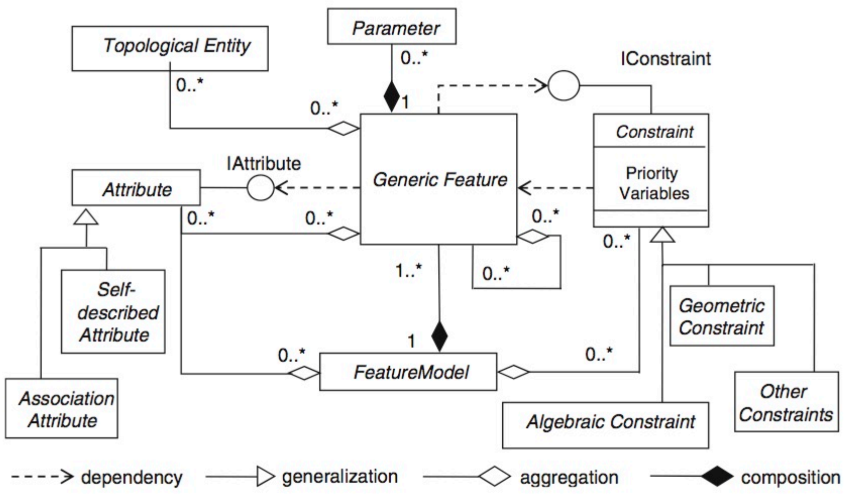


Figure 7.2: Unified Feature Model (from [TCM13])

cialises in *Self-described Attribute* and *Association Attribute* (see Fig. 7.2):

- *Self-described Attribute* covers information of various types. For example, features' (roughly speaking) *characteristics* like height, depth, diameter, surface roughness, among others, including their specific values; features' *elements* (authors' terminology), e.g., a hole's top or bottom surface; features' names, IDs, nature (whether a feature is additive or subtractive) and domain (e.g., a feature defined within a manufacturing task). However, the authors of the Unified Feature Model do not provide a complete list of information that can be covered by *Self-describedAttribute*;

- *Association attribute* is adopted to represent relationships between features, as well as features and other classes specified in a product model. For instance, an association attribute may be established between a feature and the manufacturing operations and resources required for its physical realisation on a workpiece; or, an association attribute may be adopted to establish dependencies between features and products, e.g., between a slot and the gear to which it is related.

Topological entities "are those that can be shown to the user on the screen, such as a point, line, cylinder, or cube" [TCM13, p.91].

Constraint is meant to cover geometric and algebraic relationships, among others, between the topological entities specified in a model. For example, a constraint may establish that the bottom and top surfaces of a feature are parallel. Constraints in the Unified Feature Model are associated to programming rules by which they can be, e.g., controlled throughout the designing phase. For example, a rule may be used to control the parallelism constraint between two surfaces in a feature in order to avoid running into mistakes when the model undergoes some change.

Parameters specify interlinked values between the features or topological entities in a model; e.g., a parameter may establish that if the diameter value of a hole changes, the dimension value of the shaft to be allocated in the hole has to be accordingly adjusted.[7]

Finally, *Feature Model* refers to the computer-aided model where feature information is specified.

7.2.2 Feature-based Meta-Model

The Feature-based Meta-Model proposed in [RRB15] is presented as a high-level modelling framework to manage and share feature-based data. The meta-model in UML is shown in Fig. 7.3. For the purposes of our work, we focus on *ObjectFeature* and its subclasses, since these are the core classes used to convey feature information; some of the other classes, which may be useful to understand how features are understood in the meta-model, will be just briefly presented.

Object Feature is the most general class for representing features related to objects; its subclasses – *Parameter Feature*, *Geometric Interface Feature*, *Structure Feature* and *Function Feature* – are taken from [ZS04, Rie03]. According to [RRB15], these are the most fundamental feature classes and any other feature class can be represented by specialising them with the needed information.[8]

In order to link object features to existing resources for product data management and to represent their semantics, the subclasses of *Object Feature* are subsumed by the Product-Process-Organization-Model (PPO) [NR08, ZBLDE14]. The latter is a UML (meta-)model developed to support the integrated representation of product, process and organizational knowledge for mechatronic system development. Fig. 7.4 shows the inclusion of the subclasses of *ObjectFeature* in the PPO.[9]

Accordingly, *Structure Feature* is a *PPO Component*, the latter representing the components making up assemblies. *Geometric Interface Feature* is a *PPO Interface* and represents "geometric properties through which a component can be linked to [an]other component" [RRB15, p.1131]. Basically, an interface is an entity within a product that is explicitly designed to allow for the assembly of the product; e.g., a hole designed for assembly.

[7]Parameters are used in CAx systems to facilitate the update of product models; when the dimensional values of an element in a model change, indeed, such changes often need to be propagated across the different elements defined within the model [SM95].

[8]Personal communication.

[9]Fig. 7.4 only shows the PPO classes reused in [RRB15]. The reader can refer to [NR08] for an overview of the PPO.

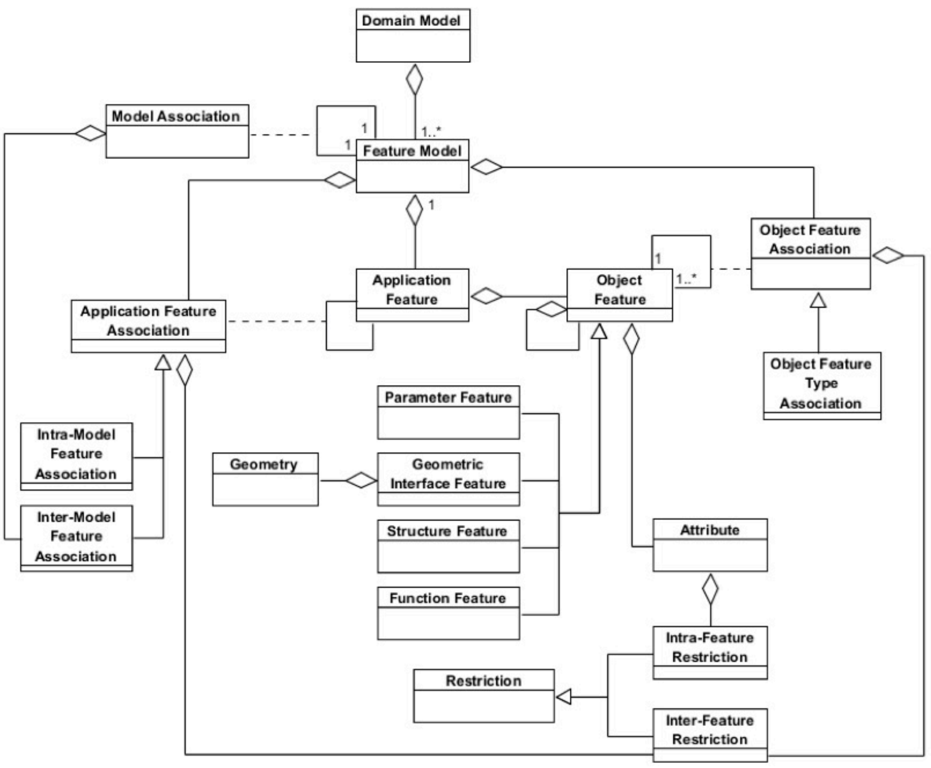

Domain Model
Model Association
1
1
1..*
Feature Model
1
Object Feature Association
1
1..*
Application Feature
Object Feature
Application Feature Association
Object Feature Type Association
Parameter Feature
Intra-Model Feature Association
Geometry
Geometric Interface Feature
Inter-Model Feature Association
Structure Feature
Attribute
Function Feature
Intra-Feature Restriction
Restriction
Inter-Feature Restriction

Figure 7.3: Feature-based Meta-Model (from [RRB15])

By '*geometric* interface feature', the authors stress that interface features are geometrically described. *Functional Feature* and *Parameter Feature* are both subsumed under *PPO Function* and refer to functional features and their parametric specification, respectively.

The class *Application Feature* (see Fig. 7.3) is meant to capture the object features specified within a specific model (*Feature Model*). Along the same lines of the Unified Feature Model [TCM13], features are specified in the Feature-based Meta-Model by means of attributes (*Attribute*), whose semantics is not however specified in [RRB15]. Relationships between features are represented by the classes *ObjectFeatureAssociation* and *ApplicationFeatureAssociation*. The latter are used to relate features within and across models, whereas the former establishes relationships between object features. The authors, however, do not provide a detailed explanation for these classes.

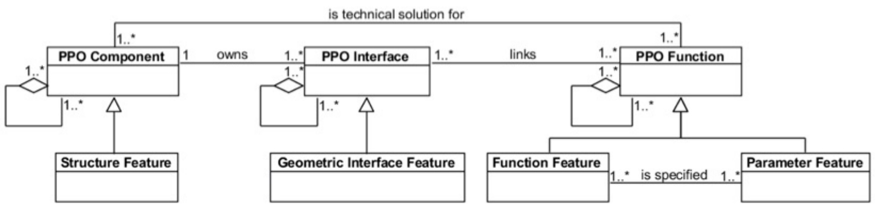


Figure 7.4: Object feature classes subsumed by the PPO (from [RRB15])

7.2.3 Critical remarks

Both the Unified Feature Model and the Feature-based Meta-Model suffer from some flaw when looked from an ontological perspective.

As regards the Unified Feature Model, the class *Attribute* covers a variety of domain entities. As said, both features' elements and characteristics (in the authors' terminology) are modelled as (UML) attributes. From an ontological perspective, "elements" and "characteristics" differ in their nature and have to be represented by different classes in order to enhance the generality, transparency and re-usability of the meta-model. For example, both a hole's surface and a hole's depth are attributes. From the previous chapters, we know that surfaces may be treated as ontological features, whereas the representation of the depth requires to distinguish between qualities, quality kinds and quality spaces. If, on the one side, the distinction between ontological features and qualities brings more classes into the picture, on the other side it is necessary to ascribe a clear semantics to data.

Also, it is not plain clear in the Unified Feature Model whether there is a semantic distinction between *Constraint* and *Association Attribute*. Take, e.g., the relationship of *parallelism* that – as we have seen – may hold between two surfaces in a feature. It is not clear why *parallelism* should be represented as a geometric constraint, rather than as an association attribute, given that both classes are meant to capture relationships. The distinction between *Constraint* and *Association Attribute* makes, however, sense from a programming perspective, given that the former is linked to a rule to check the validity of a constraint throughout the designing process. From this perspective, it seems that the Unified Feature Model mixes ontological information and the representation of the linguistic elements that are used for the specification of such information. *Association Attribute* might be understood as a class for representing relationships between features, whereas *Constraint* as a class for programming rules. If this is correct, the Unified Feature Model suffers from the ambiguity which characterises the very notion of meta-model, as we have seen at the beginning of this section. On the one hand, the Unified Feature Model is meant to capture the semantics of the notions that are used to represent and share feature-based data; on the other hand, it also represents the elements of the modelling language used for the representation. This is not a problem *per se* but a cut-off distinction has to be maintained between ontological knowledge (domain entities and their properties) and the linguistic elements used for their representation.[10]

In the Feature-based Meta-Model the semantics of object feature classes is modelled in a debatable manner with respect to the PPO. First, as we have seen, *Function Feature* and *Parameter Feature* are both subsumed by *PPO function*, which is meant to represent engineering functionalities. Differently, *Function Feature* and *Parameter Feature* are not functionalities on their own; they rather represent features that are functionally characterised and described in parametric terms, respectively. Therefore, both classes should not be subsumed under *PPO function*; rather, they should be related to *PPO function* by means of suitable relationships. Second, looking at the very notion of *Parametric Feature*, if it is meant to capture the parameters of an object feature, as it seems, an analysis of what parameters are – in an ontological sense – is required. Indeed, they are not simply dimensional values associated to dimensional qualities; as said, to model, e.g., a parameter on a feature's diameter means to model how the diameter value can change according to other elements associated to the feature. In this sense, the representation of parameters has to be considered within a larger approach for the representation of dependency relationships between features. This remains an open issue. Third, as previously said, *Geometric Interface Feature* captures (object) features that are developed to connect products. From an ontological perspective, the property of *being an interface* is functional, because it refers to the fact that a certain entity can function as connection element between two (or more) other entities. Accordingly, the notion of geometric feature has to be distinguish from the notion of interface feature, the latter being useful to functionally characterise the former with respect to assembly modelling.

In a more general perspective, both the Unified Feature Model and the Feature-based Meta-Model are not explicit about the meaning of the notion of feature for which they are meant to provide the core modelling elements. The Unified Feature Model seems to be limited to the representation and management of engineering features like slots, holes and protrusions. On the other hand, the Feature-based Meta-Model embraces a more general perspective, since it also covers components. Also, both approaches are limited to the type level, since features are represented in relation to CAx models.

[10]In the ontological analysis of product models presented in [Sect. 4.2], we informally characterise the distinction between physical entities and their design properties on the one hand, and the specification of such properties in modelling languages on the other hand.

7.2.4 Ontology-based restructuring of the meta-models

The figure below (Fig. 7.5) shows how (parts of) the meta-models previously presented can be restructured according to the ontological understanding of engineering features proposed in the previous chapters. Inasmuch as both the Unified Feature Model and the Feature-based Meta-Model represent features at the type level, we focus hereby on feature types.

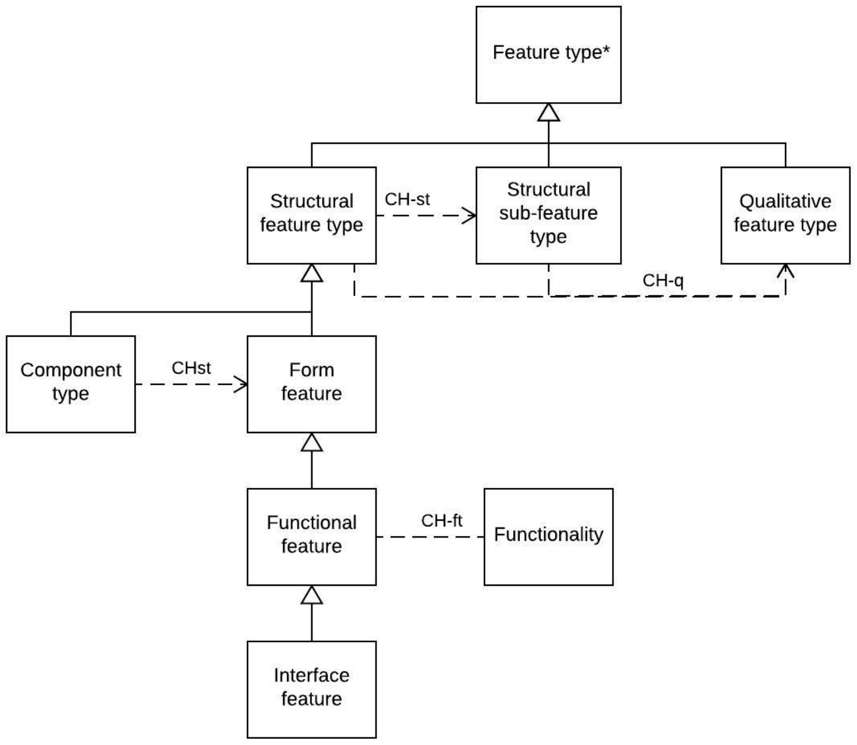


Figure 7.5: Partial restructure and integration of the meta-models in [TCM13, RRB15]

Looking at Fig. 7.5, the class *Feature type* covers structural *Structural feature type*, *Structural sub-feature type* and *Qualitative feature type.* Note that *Feature type* is marked with an asterisk in the figure; the reasons will be shortly explained.

Structural feature type – as we saw throughout the previous chapter – represents feature types that structurally characterise product types. It corresponds to the ontology-based counterpart of *Generic Feature* in [TCM13] and *Object Feature* in [RRB15], and subsumes both *Form Feature* and *Component type.*

We know from [Sect. 6.5] that *Form feature* models (at the type level) entities like holes and bosses attached to product types. For the sake of the test case, *Component type* is introduced to make sense of the *Structure Feature* class in [RRB15], which is meant to capture information about components. The relationship of structural characterisation (written as *CH-st* in Fig. 7.5) holds between *Form feature* and *Component type*, meaning that the latter is structurally characterised by the former.

Form feature is further specialised in *Functional feature*. As previously said, this captures form feature functionally characterised. This notion is not analysed in the thesis and further research is required to understand what it means for a feature to be functional and to represent it from an ontological perspective. The literature on feature-based modelling is of little help in interpreting the notion of functional feature. For the sake of the example we introduce the primitive *CH-ft* relationship between *Functional feature* and *Functionality* standing for *functional characterisation*. A functional feature might be understood as a form feature which represents what the form feature is designed for, namely, what it contributes to obtain. For example, a functional feature for assembly is used to represent that the product to which the feature is related can be assembled with another object to obtain an assembled product. As it can be seen in the figure, *Interface feature* is a subclass of *Functional feature*.

We have seen that the representation of attributes plays a central role in both the Unified Feature Model [TCM13] and the Feature-based Meta-Model [RRB15]. In [TCM13] attributes cover features' elements and characteristics (*Self-described Attribute*), as well as relationships between features and other entities specified in the models (as *Association Attribute*). On the other hand, attributes seem to cover only features characteristics in [RRB15]. We leave aside the representation of attributes as relationships, since they are not clearly represented in the meta-models. We distinguish between *Qualitative feature type* and *Structural sub-feature type*. The former class has to be used to capture qualitative information along with their quantitative values; e.g., the shape or the dimension of a hole form feature. The latter covers (structural) sub-features of (structural) feature types, e.g., the surface of a hole.[11] Note that this class is not introduced in the formal representation presented in the previous chapter, even though it can be easily represented by means of our formalism. This is the reason why the *Feature type* class is marked with an asterisk in Fig. 7.5; it is indeed meant to be differently defined from the notion of feature type in [Ch. 6] in order to cover structural sub-features, too. The relationship of *qualitative characterisation* (*CH-q* in Fig. 7.5) holds between *Structural sub-feature type*, *Structural feature type* and *Qualitative feature type*. Accordingly, instances of the latter class qualify instances of the former two classes.

Differently from the Feature-based Meta-Model, the taxonomy in Fig. 7.5 does not include *Parameter Feature*. As said, further research is required

[11]The label 'sub-feature' is borrowed from [TCM13, p.94].

on the ontological representation of parameters, possibly in terms of dependence relationships between the values of features' dimension qualities. Also, differently from the Unified Feature Model, the taxonomy does not include a *Constraint* class, since its semantics is not clear in the meta-model. Finally, differently from both the Feature-based Meta-Model and the Unified Feature Model, our approach allows for a transparent representation of the relationships between engineering features and products. As we saw in [Ch. 5], *hosting* has to be used when engineering features represent ontological features and *parthood* when they represent components; *inherence* is the relationship holding between instances of *Qualitative feature type* and the qualified objects.

Chapter 8

Conclusion

As argued across the previous chapters, the development of feature-based approaches for representing, sharing and eventually integrating multiple product models cannot avoid a systematic and unambiguous treatment of the meaning of engineering relevant notions, that of feature foremost. In order to manage data in a reliable and transparent way for both automated agents and humans, its meaning – in terms of the domain entities referred to in data models – has to be explicitly represented.

Computational ontologies and other types of information models employed for product knowledge representation currently treat features as bundles of attributes without principled approaches to distinguish features from non-features elements. Also, as we saw, they employ underspecified relationships between features, as well as between features and other elements to which they are linked, and often also combine different types of information without addressing relevant distinctions. To tackle these problems and support the development of well-founded feature-based modelling approaches, our proposal was to contextualise the treatment of feature-like notions within a broader ontological understanding of engineering knowledge. The result is a high-level ontological characterisation of engineering features.

For our purposes we adopted the core release of the DOLCE foundational ontology, namely DOLCE-CORE. This provides the basic high-level classes, relationships and axioms to represent technical knowledge. We found particularly useful the DOLCE-CORE distinction between *concepts* and *quality spaces*, distinction that we adopted and extended to represent *design properties.* In our ontology, design properties are either *design concepts* or quality spaces, possibly concretised in physical supports for communication. Design properties are thus either created (as design concepts) or adopted (as quality spaces) in designing activities to define the properties that designed objects have to satisfy when physically realised. The notion of design property allowed us to distinguish between the type and the physical levels of product knowledge representation. Among design concepts, *product types* have a special role, because they define the properties that *intentionally produced objects* have to satisfy to be *technical products.* These distinguish from both *by-products* and *defective artefacts.* The former are not intentionally produced and result as a consequence of performing production events. The latter are intentionally produced, although they fail in satisfying the design properties which are mandatory according to design experts.

Looking at engineering features, we individuated in components, qualities, amounts of matter and ontological features the high-level entities that they are meant to represent. The differences between these entities are not usually acknowledged in the engineering literature and this has led to poor conceptual frameworks. In our work the ontological treatment of qualities and amounts of matter was inherited from DOLCE-CORE. The analysis and formal representation of ontological features were proposed by taking into account the philosophical literature and previous works in the area of applied ontology, as well as by considering engineering design and manufacturing knowledge. We also proposed an ontological understanding of the notion of component. Accordingly, a component is a technical product that is explicitly designed to be assembled, even though it can exist without being related to any product. The distinction between qualities, amounts of matter, components and ontological features allows us to identify clearly the relationships holding between engineering features and the products to which they relate. Moreover, given the aforementioned distinction between the physical and the type levels of product knowledge representation, we distinguished among engineering features between *feature types* and *physical features.* The former are design concepts used to characterise either qualitatively or structurally product types; the latter are the physical entities that instantiate feature types. The distinction between feature types and physical features is common in the literature but it is often blurred, or not addressed in a systematic way.

Given the relevance of form features in modelling approaches, we analysed how they are understood across the literature. More specifically, we found that form features are commonly treated as either shape qualities, material or immaterial objects. By discussing the pros and cons of these different perspectives, we propose how they could be integrated within a unified ontological view.

Finally, note that the ontological distinctions proposed across the thesis are based on a *descriptive approach* aimed at representing in a coherent manner experts' world-views. Each notion has been indeed introduced in our work only after a careful analysis of the literature. For example, the distinction between the type and the physical levels of product knowledge representation is recurrently proposed in engineering without a systematic treatment. On the other hand, some high-level distinctions also depend on the author's personal view; for instance, we saw in [Ch. 4] that technical products are distinguished from technical roles. Even in these cases, however, we took the approach that could better fit with engineering technical knowledge.

8.1 Limits of the approach

(i) **Lack of geometric/topological information.** When feature-based approaches are adopted in CAx systems, especially in CADs, they are employed also to cover quantitative information, especially concerning the geometry or the topology of the products at hand. For example, the distinction between different form feature types like through and blind holes may be done in topological and geometric terms. Our approach is limited under this respect, since we did not provide a formal treatment of the ontology that is able to convey geometric or topological information.

However, the starting point of our research was the lack of transparency in current approaches about the domain entities that features are meant to represent. This means that before moving to a formal representation of features in a language that allows for the specification of geometric and/or topological constraints, the ontological clarification of feature-like notions is necessary. As seen in [Sect. 7.2], the adoption of geometric constraints is useful to specify that, e.g., the surfaces of a form feature are parallel. On the other hand, modelling this constraint means to assume that a form feature is something that has surfaces which can stand in a specific relationship to each other. However, this should not be taken for granted; if a form feature is, e.g., understood as a shape-quality, to be parallel are (some of) the product's surfaces where the form feature inheres, rather than the feature's surfaces. The characterisation of the ontological nature of features is therefore prior to their representation in a language suitable to catch geometric and/or topological information. Actually, the ontological understanding of features supports their geometric and/or topological treatment, because it provides an overall framework of what features are. (In the next section we comment on how our ontology may be enhanced with geometric/topological knowledge.)

(ii) **Limited coverage of feature classes.** We saw that feature-based approaches include a variety of feature classes. The ontology we proposed only covers form features and needs to be therefore extended, when further classes are required by applications. As claimed across the chapters, the choice of including the notion of form feature in the ontology relies on its relevance in feature approaches, since different feature classes are form features enriched with specific knowledge, e.g., about functionalities or manufacturing. Despite the limited coverage of feature classes, the ontology provides the overall conceptual framework to allow for the inclusion of specific classes in a principled manner.

In a broader perspective, the generality of the proposed ontology may represent a limitation when the ontology is looked from its ability to handle real-world data processed within CAx systems. As said throughout the thesis, the purpose of the presented work is to provide an ontological sound basis for feature-based product knowledge representation. As such, the ontology is not committed to specific modelling scenarios or CAx applications. Therefore, in order to function as reference ontology for engineering data management, further research work is needed for its extension. We address this issue in the next section.

(iii) **Implementation.** At the state of the art, ontologies employed within the area of product knowledge representation are mainly developed in the languages of the Semantic Web initiative, OWL and SWRL foremost. These languages are supported by tools which allow for the creation of ontologies and their population with specific data; they are also provided with reasoning mechanisms to check ontology consistency, as well as to derive theorems from the stated axioms. At the moment, our ontology is formalised in first-order logic and its representation in OWL, or another language suitable for applications is addressed as future work.

8.2 Future work

In addressing future work following up the research developed in the thesis, we distinguish between research lines concerning the further development of the presented ontology and more general issues about the interaction between applied ontology and product knowledge representation.

A) **Further developments of the presented ontology**

  (i) **Representation of the notions informally introduced in the thesis.** We presented a number of notions across the thesis distinguishing, e.g., between technical products and by-products, or between the relationships of partial and total compliance. However, only a subset of these notions were formalised to support feature modelling. Further work is necessary to axiomatise the remaining notions;

  (ii) **Complex quality spaces.** We saw in the previous chapters that the adoption of qualities spaces should allow for the representation of (complex) qualities, which combine different quality kinds, e.g., dimensions and shapes. Different strategies may be adopted to purse this line of research. Two examples are hereby given.

One the one hand, one may develop a framework for the systematic treatment of multi-dimensional quality spaces, whose interlinked dimensions allow for the integrated representation of the required information. A multi-dimensional shape space may include, e.g., a dimension for shape along with a dimension for depth and a dimension for diameter. Relationships between the dimensions are then needed to establish their mutual dependencies. For example, one may establish that a certain shape goes along with some diameter and depth values. According to this perspective, when an individual quality is located in a multi-dimensional shape space, it is located in a multi-dimensional region, which provides shape, diameter and depth values.[1]

On the other hand, one may keep separate the spaces associated to different quality kinds and model a complex individual quality as comprising qualities of different kinds. For example, a complex individual shape quality may consist in a shape quality, a depth quality and a diameter quality, where each of these qualities is located in (a region of) the space associated to its corresponding kind. As in the previous approach, dependencies between the spaces are needed to constrain the relationships between the values of the qualities.

Further research is required to firstly analyse engineering modelling cases and application requirements concerning the use of complex qualities, and to secondly investigate which ontological approach is better suited for their representation. The second approach aforementioned seems at first glance more flexible than the first one, because it does not necessarily constrain the shape space to the depth and the diameter spaces.[2]

(iii) **Enrichment of the ontology with geometric and topological knowledge.** We foresee at least two different approaches to combine our ontology with a geometric and/or topological representation of features.

First, one may enrich the formal representation of the ontology by means of a mereo-topological or mereo-geometrical formal-

[1]This is the approach proposed in [Gär14] to represent colours. The colour space is, indeed, multi-dimensional, since it comprises a dimension for hue, a dimension for brightness and a dimension of saturation. Dependencies between the dimensions establish that having a certain hue value implies having certain brightness and saturation values.

[2]We saw in [Sect. 6.5] that shapes are sometimes only qualitatively described in engineering.

ism. A number of modelling approaches based on the adoption of these formalisms have been recently proposed [GDD$^+$15, KYK08]. They nevertheless suffer from logical flaws and have not been developed on sound ontological basis. Mereo-topologies and mereo-geometries are, however, hardly tractable by computer systems, because they are developed in expressive logical languages. If the representation of topological and geometrical constraints is meant to support the computational treatment of product knowledge, a different approach may be required. (See second modelling alternative.)

Second, to trade-off the ontological representation of technical knowledge and its geometric/topological representation in a computer amenable way, one may distinguish between two different representational layers, an ontology and what may be called *feature box*. The ontology can be used for a qualitative treatment of domain knowledge that identifies and distinguishes between the domain entities at play for product knowledge representation, as well as to provide the overall vocabulary for its unambiguous treatment. The feature box, on the other side, would consist in the specification of the form features needed in an application by means of a suitable mathematical language. For example, the ontology may include a taxonomy of hole form features that distinguishes between blind, through and counterbore holes; the feature box, e.g., coded in MathLab, may provide the representation of these classes using geometric and topological formulas. A mapping system is then required to associate classes from the ontology to the formulas in the feature box.

(iv) **Specialisation of the ontology.** To foster the applicability of the ontology to specific modelling scenarios, the ontology needs to be extended with technical knowledge. This line of research may be pursued by exploring further the application of feature approaches in, e.g., CAD/CAPP integrated systems or Product Lifecycle Management (PLM) systems, and by specialising the ontology with the required classes, relationships and axioms. For example, the representation of manufacturing features may facilitate the application of the ontology to handle manufacturing data about the realisation of products in production systems; or, the representation of functional features may be used to explicitly convey functional knowledge to support, e.g., assembly modelling. The representation of both manufacturing processes and functionalities is however challenging, and requires a careful analysis of how they can be represented in a way that is both ontologically coherent and sound with respect to experts' conceptualisations.

(v) **Formalisation of the ontology for applications.** The formal representation of our ontology in a Semantic Web language would likely enhance its applicability within an information system to handle engineering data. Recall, however, that languages like OWL have been explicitly designed to trade-off expressivity and computational tractability; they are therefore less expressive than FOL. The codification of our ontology in OWL will thus limit its ability to represent experts' conceptualisations, which – as we have seen across the thesis – require the handling of complex knowledge and the use of expressive formal languages.

B) **Ontology-based product knowledge representation**

(i) **Ontology-based modelling patterns.** As argued throughout the thesis, the development of ontologies in the area of product knowledge representation is seldom guided by principled methodologies. For example, it is not uncommon to find "ontologies" that erroneously use taxonomical relationships between classes, where the intended semantics of the relationships would instead require the adoption of horizontal (non-taxonomical) links between the classes at hand. A methodology like OntoClean [GW00] is scarcely used for product knowledge representation, whereas its application can support the development of principled ontologies. Notoriously, however, OntoClean only distinguishes between different meta-properties (e.g., rigidity, identity, unity), whereas a methodology for ontology development in design or manufacturing has to support the ontological analysis and representation of more specific and recurrent modelling cases within these domains. An example that we start addressing in the thesis is how to represent the distinction between products, as physical objects, and product types, as design properties. Another example previously mentioned is how to ontologically represent qualities that combine multiple quality kinds.

To purse this line of research one should firstly look for recurrent modelling patterns for product knowledge representation; secondly, analyse the patterns from the perspective of an ontology for product knowledge representation, like the one we proposed; thirdly, represent the patterns independently from specific application platforms in order to foster their re-usability; finally, codify the patterns within a modelling application. The latter task may be carried out by means of OntoUML,[3] which provides

[3]`http://www.menthor.net/ontouml.html`, last access October 2016.

a guided methodology based on (the extension of) OntoClean and a UML tool to develop ontologies. A library of manufacturing modelling patters may be integrated in the tool to assist domain experts in the development of application ontologies. As claimed in the third bullet point of the (A)-list above, modelling tools for engineering knowledge representation should also support the integration of ontologies with (generally speaking) *boxes* to specify topological and/or geometric constraints.

(ii) **Ontological analysis and representation of relationships.** The representation of relationships between the classes in an information model is notoriously a hard task [GW08, GG15]. We have already mentioned that models developed for product knowledge representation often adopt *ad hoc* strategies in establishing relationships between the represented classes. The rigorous treatment of relationships on ontological bases has already proven its impact in improving the structure of information models, as well as their integration [SCK$^+$05, GW08]. A similar research line may be pursued for product knowledge representation aimed at individuating, analysing and representing core relationships to be adopted by various information modelling resources.

As a general and conclusive comment, it is needed in our understanding the establishment of a scientific community whose goal is to put forward the research and the application of the technics and technologies of applied ontology in the broad area of product knowledge representation. At the moment, disparate and disconnected research efforts are carried out, most of the time ignoring their respective problems and results. If different research efforts are not coordinated under an overall research agenda, it becomes hard to understand the long-terms goals that the design/manufacturing community would like to reach by the application of ontologies; consequently, it becomes hard to settle mature research initiatives that do not just aim at solving the issues of specific modelling scenarios, but contextualise these issues within a general research picture.

Appendices

Appendix A

The missing and replaceable artefact

In discussing about the ontological nature of artefacts for engineering knowledge representation, Guarino [Gua14] presents the problem of the missing and replaceable artefact. The problem goes as follow: Imagine to bring your car to the garage, because the left headlamp is damaged, and the technician who assists you observes that the car's headlamp has been already replaced twice in the last few weeks. Nothing more than an everyday scene of life, but let us ask: What is the ontological picture of this scene? More specifically, what kind of thing is the headlamp, which is meant as having been replaced twice in the car?

Apart form the thought experiment, it is common in the everyday parlance to express sentences which are meaningful only if a certain ontology is admitted. In the sentence "*The lamp* of my car has been replaced twice", it seems that the expression 'the lamp' refers to an object with its identity and unity conditions. 'The lamp', however, cannot denote a physical (and material) object, since it would be misleading to consider an object as something that can be replaced, while still preserving its own identity. The very notion of replacement indeed alludes to the fact that an object is substituted with something else. What is then 'the lamp' replaced twice in your car?

According to Guarino, the lamp is "[...] a special imaginary, conventional entity", which we shall call *conventional artefact* in the remaining of this section.[1] A conventional artefact is understood as follows in Guarino's view:

1. It is an entity that, whenever it is present in time, has a spatial location. It is therefore an object in DOLCE-CORE sense;

2. It is an artefact, because it is intentionally designed and therefore complies with a design specification;[2]

[1]Guarino talks about 'conventional component'; we use 'artefact' to be more general.

[2]As noted in [Sect. 4.3], a design specification for Guarino establishes the properties

3. It is constantly specifically dependent for its existence on another artefact, called its *host*. For example, the conventional lamp depends on the car, which is thus the host of the conventional lamp. Note that Guarino explicitly stresses that the dependence has to be on a *specific* artefact, which means that the conventional left headlamp of *my* car exists as long as *my* car exists. (Recall the difference between generic and specific dependence mentioned in [Sect. 6.2].)

4. It is present in time in two different ways, namely:

   (a) Actual presence: a conventional artefact is *actually present* at a certain time t in the physical location r established by the corresponding design specification ds, if there is a physical object that is located in r and that complies with ds. The physical object is called the *physical constituent* of the conventional artefact. For example, the conventional left headlamp $cv\_l\#$ of my car is actually present in the dedicated location $r\#$ of my car established by the design specification $ds\#$, when there is a physical lamp $ph\_l\#$ located in $r\#$ that complies with $ds\#$. (The latter constraint guarantees that in order for a physical object to be "attached to" an artefact, it has to meet some pre-defined design properties.)

   (b) Virtual presence: a conventional artefact is virtually present at a certain time t in the physical location established by the design specification, if it is not actually present at t but its host is present at t. Accordingly, when the conventional left headlamp $cv\_l\#$ of my car is virtually present at t in the dedicated location $r\#$ of the car established by the design specification $ds\#$, this means the car is present at t but there is no physical lamp $ph\_l\#$ located in $r\#$ at t.

5. It is characterised by a number of nominal qualities. When it is actually present, a conventional artefact inherits the properties of its constituent. Guarino provides the following example: "[...] when a lamp different from the recommended one is mounted [in a car], we can say that the headlamp has a nominal power of 35W, while the actual power is, say, 50 W".

Following this line of reasoning, Guarino himself recognises that a conventional object is some sort of ontological feature (in DOLCE sense). More specifically, he is inclined to see a conventional object as a *relative place* (see [Sect. 5.6]) in an artefact that is empty in the case of virtual presence and filled with matter (in his terminology) when actually present.

that some physical object has to satisfy to be considered as a realisation of the specification. A design specification can be a mental entity in a designer's head.

To summarise, in order to make ontologically sense of the missing and replaceable artefact problem, Guarino proposes to consider (at least) the following entities: design specifications, physical artefacts, conventional artefacts and their physical constituents. We argue in the following that Guarino's notion of conventional artefact is both problematic and unnecessary. We show how the ontological theory presented in the previous chapters can deal with the missing and replaceable artefact problem by getting rid altogether of conventional objects.

Our main concern with the notion of conventional object is the idea that it is a physical object which can be actually present at some time or only virtually present at some other time. Assuming Guarino's perspective, let us consider John's car *car#* whose left physical headlamp was removed at a certain time t from the car's location $r\#$. Following Guarino, even if a physical headlamp is not in place, there is still the conventional headlamp located in $r\#$. Also, as long as a physical headlamp is not there, the conventional headlamp is only virtually present, but once a physical headlamp is placed in $r\#$, the conventional headlamp turns into actual presence.

What does it mean that a conventional object can turn from pure virtual to actual presence and vice versa? It seems that a conventional object is sometimes immaterial and sometimes material. In some case, indeed, it is present in space and time as something immaterial (virtual presence), whereas in some other case it is present "in flesh and blood" in virtue of the presence of a physical object by which it is constituted (actual presence). But what kind of object can be either immaterial or material at different times while preserving its own identity? The idea that an object can completely loose and acquire materiality (and vice versa) while remaining the same is hard to swallow. Apart from this and other ontological concerns that may be raised,[3] even from a common sense perspective it seems unnatural to think that talking about the lamp of my car, when there is no lamp in the car, means referring to some object, which is in the car but in an immaterial way. It is true that a technician may (ambiguously) pronounce the sentence "The lamp has been changed twice this year", as if he or she referred to a specific object, but conventional objects demand more of what they are meant to solve.

Moving now to the *pars construens* of the discussion, we argue that the notion of design concept along with an overall understanding of physical artefacts is enough to face the missing and replaceable artefact problem. Consider once again the example: John's car is there at t with the left headlamp removed, that is, there is no left (physical) headlamp in the car at

[3]Just to briefly mention another issue, it remains unclear what it means that a conventional object inherits the qualities of the constituting object. Note that if a conventional object is designed, it may not inherit (whatever it means) qualities that are not defined in the corresponding design (specification).

t. Then, John brings the car to the garage where it gets its new left headlamp at t'. This new lamp, call it *ph_l#1*, can be installed in (a component of) the car just because it complies with the product type of the car. The product type, indeed, establishes that the headlamps to be installed in the car need to have, e.g., a certain dimension and voltage. To be more precise, in our theory we would say that *ph_l#1* has to comply with the (design) type that structurally characterises the product type of the car. In other words, there are some design properties *dp#* within the product type *pt#* that enjoy identity and unity conditions, and that have to be satisfied by physical lamps, in order for the latter to be installed in the car. Accordingly, when a physical lamp is *missing* from the car, this means that there is no physical object located in the dedicated location of the car that satisfies the type *dp#*; when the physical lamp *ph_l#1* is *replaced* with another lamp *ph_l#2*, then both of them complies (at least at certain times) with *dp#* (in a relevant way).

In this picture there is no object within a product that is either virtually or actually present, whenever a product's component is missed or replaced. There are artefacts which can be substituted within larger artefacts on the basis of their design properties. From the perspective of a common sense statement like "The lamp of my car has been already replaced twice", its surface semantics should not bring us into ontological quagmires. One may assume that common sense is sometimes loose in its way of speaking. The expression 'the lamp' in the sentence above may, indeed, refer to the physical objects that were placed in and removed from the car. The sentence above only apparently refers to a single object, whereas it refers to two different lamps, which satisfy some common design properties. The sentence may not represent a coherent way of speaking with respect to an underlying ontology of physical objects and artefacts, but we are not forced to get common sense seriously from an ontological perspective.

To conclude, Guarino succeeds in the semantic analysis of common sense sentences, when the latter seem to refer to objects that can miss from or can be replaced within other objects while preserving their own identity. However, his ontology brings into the picture conventional objects as entities whose identity conditions raise some doubts, since they are meant as being material at some times and immaterial at some other times. On the other hand, our ontology faces the missing and replaceable artefact problem while getting rid of conventional objects, by relying on the distinction between design properties and the artefacts that comply with these properties. This should not be seen as an Occamist strategy that simply cuts away the problem; it is rather a different approach to deal with it. Accordingly, the replacement of an artefact a_1 with another artefact a_2 within a larger artefact a_3 is done on the basis of the (design) type *ty* that both a_1 and a_2 satisfy (in some relevant way). When an artefact a_1 is missing from a larger artefact

a_3, it means that there is no artefact in a_3 that complies with ty, namely, with the type that a_1 would satisfy, were it included in a_3. This approach may, however, require to rephrase common sense sentences in order to get rid of their superficial semantics. As a methodological remark, this does not mean that common sense is completely misguided in its ontological claims; it rather means that its ambiguity is a source of traps for the ontologist.

Appendix B

Glossary of terms

ARTEFACT	**Artefact**
DESIGNCONCEPT	**Design Concept**
FEATURETYPE$_Q$	**Qualitative feature type**; FEATURETYPE$_Q(c_1, c_2)$, "c_1 is a qualitative feature type in c_2"
FEATURETYPE$_{ST}$	**Structural feature type**; FEATURETYPE$_{ST}(c_1, c_2)$, "c_1 is a structural feature type in c_2"
FEATURETYPE	**Feature type**; FEATURETYPE(x, y), "x is a feature type in y"
PHYSICALFEATURE$_Q$	**Qualitative physical feature**; PHYSICALFEATURE$_Q(q, x)$, "q is a qualitative ph-feature in x"
PHYSICALFEATURE$_{ST}$	**Structural physical feature**; PHYSICALFEATURE$_{ST}(x, y)$, "x is a structural ph-feature in y"
PHYSICALFEATURE	**Physical feature**; PHYSICALFEATURE(x, y), x is a physical feature in y
FORMFEATURE	**Form feature**; FORMFEATURE(x, y), "x is a form feature in y

Table B.1: Glossary

F	**Ontological feature**
MaterialFeature	**Ontological material feature**
ImmaterialFeature	**Ontological immaterial feature**
MaterialObject	**Material object**
ImmaterialObject	**Immaterial object**
ProductType	**Product type**
TechComponent	**Technical component**
TechProduct	**Technical product**

Table B.2: Glossary cont'd

CN	**Concept**
E	**Event**
M	**Amount of matter**
OB	**Object**
PT	**Particular**
Q	**Individual quality**
SP	**Quality space**

Table B.3: Glossary of DOLCE-CORE classes

Bibliography

[ABAS16] A. Abadi, H. Ben-Azza, and S. Sekkat. An ontology-based framework for virtual enterprise integration and interoperability. In *Electrical and Information Technologies (ICEIT), 2016 International Conference on*, pages 36–41. IEEE, 2016.

[Adr13] P. Adriaans. Information. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2013 edition, 2013.

[AGGS$^+$14] S. Abdul-Ghafour, P. Ghodous, B. Shariat, E. Perna, and F. Khosrowshahi. Semantic interoperability of knowledge in feature-based CAD models. *Computer-Aided Design*, 56:45–57, 2014.

[AGGSP07] S. Abdul-Ghafour, P. Ghodous, B. Shariat, and E. Perna. A common design-features ontology for product data semantics interoperability. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 443–446. IEEE Computer Society, 2007.

[AH15] F. Ahmed and S. Han. Interoperability of product and manufacturing information using ontology. *Concurrent Engineering*, 23(3):265–278, 2015.

[AHC15] M.M. Andreasen, Claus T. Hansen, and P. Cash. *Conceptual Design: Interpretations, Mindset and Models*. Springer, 2015.

[AHYC12a] N. Anjum, J. A. Harding, R.I.M. Young, and K. Case. Manufacturability Verification through Feature-based Ontological Product Models. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 22, 2012.

[AHYC12b] N. Anjum, J.A. Harding, R.I.M. Young, and K. Case. Mediation of foundation ontology based knowledge sources. *Computers in Industry*, 63(5):433–442, 2012.

[AK07] S.M. Amaitik and S.E. Kiliç. An intelligent process planning system for prismatic parts using STEP features. *The International Journal of Advanced Manufacturing Technology*, 31(9-10):978–993, 2007.

[Ale79] C. Alexander. *The timeless way of building*. Oxford University Press, 1979.

[And92] M.M. Andreasen. Designing on a designer's workbench (DWB). In *Proceedings of the 9th WDK Workshop*, pages 233–249, 1992.

[Anj11] N. Anjum. *Verification of Knowledge shared across Design and Manufacture using a Foundation Ontology*. PhD thesis, Loughborough University, 2011.

[AS09] S. Ahmed and M. Storga. Merged ontology for engineering design: Contrasting empirical and theoretical approaches to develop engineering ontologies. *AI EDAM*, 23(4):391–407, 2009.

[ASS15] R. Arp, B. Smith, and A.D. Spear. *Building Ontologies with Basic Formal Ontology*. Mit Press, 2015.

[BB00] R. Bidarra and W.F. Bronsvoort. Semantic feature modelling. *Computer-Aided Design*, 32(3):201–225, 2000.

[BBB00] M. Belaziz, A. Bouras, and J-M. Brun. Morphological analysis for product design. *Computer-Aided Design*, 32(5):377–388, 2000.

[BC14] M. Bertamini and R. Casati. Figures and holes. In J. Wagemans, editor, *Handbook of Perceptual Organization*, pages 281–293. Oxford University Press, 2014.

[BCGV09] S. Borgo, M. Carrara, P. Garbacz, and P.E. Vermaas. A formal ontological perspective on the behaviors and functions of technical artifacts. *AI EDAM*, 23:3–21, 1 2009.

[BCM$^+$03] F. Baader, D. Calvanese, D.L. McGuinnes, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BFG$^+$11] S. Borgo, M. Franseen, P. Garbacz, Y. Kitamura, R. Mizoguchi, and P.E. Vermaas. Technical artifact: An integrated perspective. In P.E. Vermaas and V. Dignum, editors, *Formal Ontologies Meet Industry. Proceedings of the 5th International Workshop (FOMI)*, pages 3–15. IOS Press, Amsterdam, 2011.

[BFG+14] S. Borgo, M. Franssen, P. Garbacz, Y. Kitamura, R. Mizoguchi, and P.E. Vermaas. Technical artifacts: An integrated perspective. *Applied ontology Journal*, 9(3-4):217–235, 2014.

[BG00] G. Brunetti and B. Golob. A feature-based approach towards an integrated product model including conceptual design information. *Computer-Aided Design*, 32(14):877–887, 2000.

[BG05] G. Brunetti and S. Grimm. Feature ontologies for the explicit representation of shape semantics. *International Journal of Computer Applications in Technology*, 23(2):192–202, 2005.

[BGM96] S. Borgo, N. Guarino, and C. Masolo. Qualitative spatial modelling based on parthood, strong connection and congruence. Internal report, Laboratory for Applied Ontology (ISTC-CNR), 1996.

[BJ93] W.F. Bronsvoort and F.W. Jansen. Feature modelling and conversion. key concepts to concurrent engineering. *Computers in Industry*, 21(1):61–86, 1993.

[BL07] S. Borgo and P. Leitao. Foundations for a core ontology of manufacturing. In R. Sharman, R. Kishore, and R. Ramesh, editors, *Ontologies. A Handbook of Principles, Concepts and Applications in Information Systems*, pages 751–775. Springer US, 2007.

[BM10] S. Borgo and C. Masolo. Full mereogeometries. *The review of symbolic logic*, 3(4):521–567, 2010.

[BM13] S. Borgo and C. Masolo. Foundational choices in DOLCE. In S. Staab and R. Studer, editors, *Handbook on Ontologies.* Springer Science & Business Media, 2013.

[BN00] M.P. Bhandarkar and R. Nagi. STEP-based feature extraction from STEP geometry for agile manufacturing. *Computers in Industry*, 41:3–24, 2000.

[BNM08] B. Babic, N. Nesic, and Z. Miljkovic. A review of automated feature recognition with rule-based pattern recognition. *Computers in Industry*, 59:321–337, 2008.

[Boo94] G. Boothroyd. Product design for manufacture and assembly. *Computer-Aided Design*, 26(7):505–520, 1994.

[Bor14] S. Borgo. An ontological approach for reliable data integration in the industrial domain. *Computers in Industry*, 65(9):1242–1252, 2014.

[Bro03] D.C. Brown. Functional, behavioral and structural features. In *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 895–900. American Society of Mechanical Engineers, 2003.

[Bru03] G. Brunetti. Feature-based virtual engineering. In R. Soenen and G.J. Olling, editors, *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, 2003.

[BS11] G. Baxter and I. Sommerville. Socio-technical systems: From design methods to systems engineering. *Interacting with computers*, 23(1):4–17, 2011.

[BSŠT14] S. Borgo, E.M. Sanfilippo, A. Šojić, and W. Terkaj. Towards an ontological grounding of IFC. In A. Rademaker and V.K. Chaudhri, editors, *Proceedings of the 6th Workshop on Formal Ontology Meets Industry (FOMI)*, volume 1333. Ceur workshop proceedings, 2014.

[BSŠT15] S. Borgo, E.M. Sanfilippo, A. Šojić, and W. Terkaj. Ontological analysis and engineering standards: An initial study of IFC. In *Ontology Modeling in Physical Asset Integrity Management*, pages 17–43. Springer, 2015.

[BV09] S. Borgo and L. Vieu. Artefacts in formal ontology. In A. Meijers, editor, *Handbook of Philosophy of Technology and Engineering Sciences*, pages 273–308. Elsevier, 2009.

[BVLDV09] J. Beetz, J. Van Leeuwen, and B. De Vries. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(01):89–101, 2009.

[BZSL10] C. Bock, X-F. Zha, H-W. Suh, and J-H. Lee. Ontological product modeling for collaborative design. *Advanced Engineering Informatics*, 24(4):510–524, 2010.

[CB14] A. Chakrabarti and L.T.M. Blessing. *An Anthology of Theories and Models of Design: Philosophy, Approaches and Empirical Explorations.* Springer Science & Business Media, 2014.

[CCF+09] C.E. Catalano, E. Camossi, R. Ferrandes, V. Cheutet, and N. Sevilmis. A product design ontology for enhancing shape processing in design workflows. *Journal of Intelligent Manufacturing*, 20(5):553–567, 2009.

[Chu10] N. Chungoora. *A Framework to Support Semantic Interoperability in Product Desing and Manufacture*. PhD thesis, Loughborough University, 2010.

[CMS07] G. Colombo, A. Mosca, and F. Sartori. Towards the design of intelligent CAD systems: An ontological approach. *Advanced Engineering Informatics*, 21(2):153–168, 2007.

[CMTT04] G. Chen, Y-S. Ma, G. Thimm, and S-H. Tang. Unified feature modeling scheme for the integration of CAD and CAx. *Computer-Aided Design and Applications*, 1(1-4):595–601, 2004.

[CMTT06] G. Chen, Y-S. Ma, G. Thimm, and S-H. Tang. Associations in a unified feature modeling scheme. *Journal of Computing and Information Science in Engineering*, 6(2):114–126, 2006.

[CMV03] O. Coma, C. Mascle, and P. Veron. Geometric and form feature recognition tools applied to a design for assembly methodology. *Computer-Aided Design*, 35:1193–1210, 2003.

[Coc91] N. Cocchiarella. Formal ontology. In H. Burkhardt and B. Smith, editors, *Handbook of Metaphysics and Ontology*, pages 640–647. Philosophia Verlag, 1991.

[CR08] A.G. Cohn and J. Renz. Qualitative spatial representation and reasoning. In F. Van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, pages 551–596. Elsevier, 2008.

[CRS$^+$13] S.K. Chandrasegaran, K. Ramani, R.D. Sriram, I. Horváth, A. Bernard, R.F. Harik, and W. Gao. The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design*, 45(2):204–228, 2013.

[CRT10] X. Chang, R. Rai, and J. Terpenny. Development and utilization of ontologies in design for manufacturing. *Journal of Mechanical Design*, 132(2):1–12, 2010.

[CTCM13] Z. Cheng, S-H. Tang, G. Chen, and Y-S. Ma. Unified feature paradigm. In Y-S. Ma, editor, *Semantic Modeling and Interoperability in Product and Process Engineering*, pages 117–142. Springer, 2013.

[CV94] R. Casati and A. Varzi. *Holes and other Superficialities.* Mit Press, 1994.

[CV99] R. Casati and A. Varzi. *Parts and Places: The Structures of Spatial Representation.* MIT Press, 1999.

[CV14] R. Casati and A. Varzi. Holes. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2014 edition, 2014.

[CY11] N. Chungoora and R. I.M. Young. Semantic reconciliation across design and manufacturing knowledge models: A logic-based approach. *Applied Ontology*, 6(4):295–315, 2011.

[CYG+13] N. Chungoora, R.I.M. Young, G. Gunendran, C. Palmer, Z. Usman, N. Anjum, A.F. Cutting-Decelle, J.A. Harding, and K. Case. A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, 64(4):392–401, 2013.

[DD15] I. Dragos and M.R. Dijmarescu. Configuration management by DOLCE upper-level ontology. In *7th International Conference on Manufacturing Science and Education*, 2015.

[DEBB05] L.M. Deshayes, O. El Beqqali, and A. Bouras. The use of process specification language for cutting processes. *International Journal of Product Development*, 2(3):236–253, 2005.

[Den99] D. Deneux. Introduction to assembly features: An illustrated synthesis methodology. *Journal of Intelligent Manufacturing*, 10(1):29–39, 1999.

[DFEG+90] T.L. De Fazio, A.C. Edsall, R.E. Gustavson, J.A. Hernandez, P.M. Hutchins, H.W. Leung, S.C. Luby, R.W. Metzinger, J.L. Nevins, and K.K. Tung. A prototype of feature-based design for assembly. *Advances in Design Automation*, 1, 1990.

[DGG+07] C. Dartigues, P. Ghodous, M. Grüninger, D. Pallez, and R. Sriram. CAD/CAPP integration using feature ontology. *Concurrent Engineering*, 15(2):237–249, 2007.

[Dip93] R.R. Dipert. *Artifacts, art works, and agency.* Temple, 1993.

[DMK12] F. Demoly, A. Matsokis, and D. Kiritsis. A mereotopological product relationship description approach for assembly oriented design. *Robotics and Computer-Integrated Manufacturing*, 28(6):681–693, 2012.

[Don04] M. Donnelly. On parts and holes: the spatial structure of the human body. In *Proceedings of the 11th World Congress on Medical Informatics (MedInfo-04)*, pages 351–356, 2004.

[Don05] M. Donnelly. Relative places. *Applied Ontology*, 1(1):55–75, 2005.

[DP94] D.Schenck and P.Wilson. *Information Modeling: the EXPRESS way.* Oxford University Press, 1994.

[DPD16] D. Decriteau, J-P. Pernot, and M. Daniel. Towards a declarative modeling approach built on top of a CAD modeler. *Computer-Aided Design and Applications*, pages 1–10, 2016.

[DPZ02] M. Dahchour, A. Pirotte, and E. Zimanyi. Materialization and its metaclass implementation. *IEEE Transactions on knowledge and data engineering*, 14(5):1078–1094, 2002.

[DS13] M. Deja and M.S. Siemiatkowski. Feature-based generation of machining process plans for optimised parts manufacture. *Journal of Intelligent Manufacturing*, 24(4):831–846, 2013.

[DSBDA04] P. Di Stefano, F. Bianconi, and L. Di Angelo. An approach for feature semantics recognition in geometric models. *Computer-Aided Design*, 36(10):993–1009, 2004.

[DWGB04] C.W. Dankwort, R. Weidlich, B. Guenther, and J.E. Blaurock. Engineers' CAx education– it's not only CAD. *Computer-Aided Design*, 36(14):1439–1450, 2004.

[EH08] W.E. Eder and S. Hosnedl. *Design Engineering: A Manual for Enhanced Creativity.* CRC Press, 2008.

[EKGT$^+$16] S. El Kadiri, B. Grabot, K-D. Thoben, K. Hribernik, C. Emmanouilidis, G. von Cieminski, and D. Kiritsis. Current trends on ICT technologies for enterprise information systems. *Computers in Industry*, 79:14–33, 2016.

[EKK$^+$13] K. Eum, M. Kang, G. Kim, M.W. Park, and J.K. Kim. Ontology-based modeling of process selection knowledge for machining feature. *International Journal of Precision Engineering and Manufacturing*, 14(10):1719–1726, 2013.

[EKK15] S. El Kadiri and D. Kiritsis. Ontologies in the context of product lifecycle management: state of the art literature review. *International Journal of Production Research*, 53(18):5657–5668, 2015.

[ElM91] H.A. ElMaraghy. Intelligent Product Design and Manufacture. In D.T. Pahm, editor, *Artificial Intelligence in Design*, pages 147–168. Springer Verlag London, 1991.

[FCG^{+}15] S.R. Fiorini, J.L. Carbonera, P. Gonçalves, V.A.M. Jorge, V.F. Rey, T. Haidegger, M. Abel, S.A. Redfield, S. Balakirsky, V. Ragavan, et al. Extensions to the core ontology for robotics and automation. *Robotics and Computer-Integrated Manufacturing*, 33:3–11, 2015.

[FFBS08] S.J. Fenves, S. Foufou, C. Bock, and R.D. Sriram. CPM: a core model for product data. *Journal of Computing and Information Science in Engineering*, 8(1), 2008.

[FG08] R. Ferrario and N. Guarino. Towards an ontological foundation for services science. In *Future Internet Symposium*, pages 152–169. Springer, 2008.

[FGL^{+}07] X. Fiorentini, I. Gambino, V-C. Liang, S. Rachuri, M. Mani, C. Bock, C.M. Gutierrez, and J.M. Turner. An ontology for assembly representation. 7436, NIST, 2007.

[FGM99] M. Fontana, F. Giannini, and M. Meirana. A free form feature taxonomy. In *Computer Graphics Forum*, volume 18, pages 107–118, 1999.

[Flu03] V. Flusser. *Filosofia del design.* Mondadori Bruno, 2003.

[FOL^{+}03] M.W. Fu, S-K. Ong, W.F. Lu, I.B.H. Lee, and A.Y.C. Nee. An approach to identify design and manufacturing features from a data exchanged part model. *Computer-Aided Design*, 35(11):979–993, 2003.

[fSI94] International Organization for Standardization (ISO). *Industrial Automation Systems and Integration - Product Data Representation and Exchange. Part 1: Overview and fundamental principles.* ISO Geneve, 1994.

[fSI04] International Organization for Standardization (ISO). *Industrial Automation Systems and Integration - Physical device control - Data model for computerized numerical controllers - Part 10: General Process Data.* ISO Geneve, 2004.

[fSI06] International Organization for Standardization (ISO). *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 224, Mechanical Product Definition for Process Planning Using Machining Features.* ISO Geneve, 2006.

[Gal97] P. Galle. In defense of types in knowledge-based CAAD. In J. Pohl, editor, *Advances in Collaborative Design and Decision-Support Systems*, pages 111–119. CAD Research Center, Cal Poly, 1997.

[Gal98] P. Galle. Design as intentional action: A conceptual analysis. *Design studies*, 20(1):57–81, 1998.

[Gal00] A. Galton. *Qualitative Spatial Change.* Oxford University Press, 2000.

[Gal08] P. Galle. Candidate worldviews for design theory. *Design Studies*, 29(3):267–303, 2008.

[Gär04] P. Gärdenfors. *Conceptual spaces: The geometry of thought.* MIT press, 2004.

[Gär14] P. Gärdenfors. *The Geometry of Meaning: Semantics based on Conceptual Spaces.* MIT Press, 2014.

[Gar15] W. Gareth. *Design: An essential Introduction.* Goodman-Fiell, 2015.

[Gar16] P. Garbacz. A formal ontology of texts. In Roberta Ferrario and Werner Kuhn, editors, *Formal Ontology in Information Systems. Proceedings of the 9th International Conference (FOIS).*, volume 283, pages 345–358. IOS Press, 2016.

[GB00] N. Giralt and P. Bloom. How special are objects? children's reasoning about objects, parts, and holes. *Psychological Science*, 11(6):497–501, 2000.

[GBM97] N. Guarino, S. Borgo, and C. Masolo. Logical modelling of product knowledge: Towards a well-founded semantics for STEP. In *Proceedings of European Conference on Product Data Technology*, pages 183–190. Citeseer, 1997.

[GDD$^+$15] E. Gruhier, F. Demoly, O. Dutartre, S. Abboudi, and S. Gomes. A formal ontology-based spatiotemporal mereotopology for integrated product design and assembly sequence planning. *Advanced Engineering Informatics*, 29(3):495–512, 2015.

[Ger90] J.S. Gero. Design prototypes: A knowledge representation schema for design. *AI magazine*, 11(4), 1990.

[GF$^+$92] M.R. Genesereth, R.E. Fikes, et al. Knowledge interchange format-version 3.0: Reference manual. Technical report, Computer Science Department, Stanford University Stanford, California, USA, 1992.

[GG95] P. Giaretta and N. Guarino. Ontologies and knowledege bases: Towards a terminological clarification. *Towards Very Large Knowldege Bases: Knowldege Building and Knowledge Sharing*, 25, 1995.

[GG15] N. Guarino and G. Guizzardi. "We need to discuss the relationship": Revisiting relationships as modeling constructs. In *International Conference on Advanced Information Systems Engineering*, pages 279–294. Springer, 2015.

[GM05] N. Guarino and M.A. Musen. Applied ontology: Focusing on content. *Applied Ontology*, 1(1):1–5, 2005.

[GOP02] M.P. Gallaher, A.C. O'Connor, and T. Phelps. Economic impact assessment of the international standard for the exchange of product model data (STEP) in transportation equipment industries. Technical Report Planning report 02-5, NIST, 2002.

[GOS13] N. Guarino, D. Oberle, and S. Staab. What is an Ontology? In S. Staab and R. Studer, editors, *Handbook on ontologies*, pages 1–17. Springer Science & Business Media, 2013.

[GPFLC04] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering with Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web.* Springer Verlag Berlin Heidelberg, 2004.

[Gro07] M.P. Groover. *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems.* John Wiley & Sons, 2007.

[Gru93] T. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[Grü09] M. Grüninger. Using the PSL ontology. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 423–443. Springer-Verlag Berlin Heidelberg, 2009.

[GSM15] N. Guarino and M.R. Stufano Melone. On the ontological status of design objects. In F.A. Lisi and S. Borgo, editors, *Proceedings of the 1st Workshop on Artificial Intelligence & Design (AIDE)*, volume 1473, pages 27–32. CEUR, 2015.

[Gua94] N. Guarino. The ontological level. In R. Casati, B. Smith, and G. White, editors, *Philosophy and the Cognitive Sciences.* Hölder-Pichler Tempsky, 1994.

[Gua95] N. Guarino. Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human-Computer studies*, 43(5):625–640, 1995.

[Gua98] N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Formal Ontology in Information Systems. Proceedings of the First International Conferece (FOIS)*, pages 3–15. IOS Press, 1998.

[Gua14] N. Guarino. Artefactual systems, missing components and replaceability. In M. Franseen, P. Kroes, T. Reydon, and P.E. Vermaas, editors, *Artefact Kinds. Ontology and the Human-Made World*, pages 191–206. Springer-Verlag Berlin Heidelberg, 2014.

[Gui05] G. Guizzardi. *Ontological Foundations for Structural Conceptual Models*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2005.

[GV99] B. Giovanni and P. Vidali. *Filosofia della scienza*. Bruno Mondadori, 1999.

[GW00] N. Guarino and C. Welty. Towards a methodology for ontology-based model engineering. In *Proceedings of the ECOOP 2000. Workshop on Model Engineering, Cannes, France*, 2000.

[GW08] G. Guizzardi and G. Wagner. What's in a relationship: An ontological analysis. In *International Conference on Conceptual Modeling*, pages 83–97. Springer, 2008.

[Hah13] T. Hahmann. *A Reconciliation of Logical Representations of Space: From Multidimensional Mereotopology to Geometry*. PhD thesis, University of Toronto, 2013.

[Han96] J.H. Han. Survey of feature research. Technical Report IRIS-96-346, Institute for Robotics and Intelligent Systems, USC,USA, 1996.

[HB12] T. Hahmann and B. Brodaric. The void in hydro ontology. In M. Donnelly and G. Guizzardi, editors, *Formal Ontology in Information Systems. Proceedings of the 7th International Conference (FOIS).*, volume 239, pages 45–58, 2012.

[Hed04] S. Hedman. *A First Course in Logic*. Oxford University Press Oxford, 2004.

[HF06] M. Hou and T.N. Faddis. Automatic tool path generation of a feature-based CAD/CAPP/CAM integrated system. *International Journal of Computer Integrated Manufacturing*, 19(4):350–358, 2006.

[HH06] H. Herre and B. Heller. Semantic foundations of medical information systems based on top-level ontologies. *Knowledge-Based Systems*, 19(2):107–115, 2006.

[HHPS13] A.S.M. Hoque, P.K. Halder, M.S. Parvez, and T. Szecsi. Integrated manufacturing features and design-for-manufacture guidelines for reducing product cost under CAD/CAM environment. *Computers & Industrial Engineering*, 66(4):988–1003, 2013.

[Hil93] R. Hilpinen. Authors and artifacts. In *Proceedings of the Aristotelian Society*, volume 93, pages 155–178. JSTOR, 1993.

[Hil11] R. Hilpinen. Artifact. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2011 edition, 2011.

[HKR09] P. Hitzler, M. Krotzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. CRC Press, 2009.

[HLGF10] O. Hamri, J-C. Léon, F. Giannini, and B. Falcidieno. Software environment for CAD/CAE integration. *Advances in Engineering Software*, 41(10):1211–1222, 2010.

[Hor04] I. Horvath. A treatise on order in engineering design research. *Research in Engineering Design*, 15(3):155–181, 2004.

[HPR00] J. Han, M. Pratt, and W.C. Regli. Manufacturing feature recognition from solid models: A status report. *IEEE Transactions on Robotics and Automation*, 16(6):782–796, 2000.

[HPSB$^+$04] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 2004.

[HS12] B. Henderson-Sellers. *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages*. Springer Science & Business Media, 2012.

[HT96] I. Horvath and V. Thernesz. Morphology-inclusive conceptual modeling with feature objects. In Y. Shuzi, Z. Ji, and L. Cheng-Gang, editors, *Proc. SPIE 2644, Fourth International Conference on Computer-Aided Design and Computer Graphics*, pages 563–572, 1996.

[HV09] W. Houkes and P.E. Vermaas. Produced to use. *Techné: Research in Philosophy and Technology*, 13(2):123–136, 2009.

[Imr13] M. Imran. *Towards an Assembly Reference Ontology for Assembly Knowledge Sharing.* PhD thesis, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, 2013.

[Int16] BuildingSmart International. *Industry Foundation Classes (IFC), version 4 - addendum 1*, 2016.

[I38] International Standards Organization (ISO). *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 203: Application Protocol, Configuration controlled Design 3D Designs of Mechanical Parts and Assemblies.* ISO Geneve, 2004.

[IY15] M. Imran and R.I.M. Young. The application of common logic-based formal ontologies to assembly knowledge sharing. *Journal of Intelligent Manufacturing*, 26(1):139–158, 2015.

[JGAB09] J.T. Jagenberg, E.A. Gilsdorf, R. Anderl, and T. Bornkessel. Knowledge driven design features for the product life cycle of engine parts. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 721–728. American Society of Mechanical Engineers, 2009.

[Kas10] G. Kassel. A formal ontology of artefacts. *Applied Ontology*, 5(3-4):223–246, 2010.

[KBF$^+$09] S. Krima, R. Barbau, X. Fiorentini, R. Sudarsan, and R.D. Sriram. OntoSTEP: OWL-DL Ontology for STEP. In *Proceedings of 6 th International Conference on Product Lifecycle Management*, 2009.

[Kee11] C.M. Keet. The use of foundational ontologies in ontology development: An empirical assessment. In *The Semantic Web: Research and Applications*, pages 321–335. Springer, 2011.

[Kro12] P. Kroes. *The dual nature of technical artefacts: Creations of Mind and Matter. A Philosophy of Engineering Design.* Springer, 2012.

[KWMN04] K-Y. Kim, Y. Wang, O.S. Muogboh, and B.O. Nnaji. Design formalism for collaborative assembly design. *Computer-Aided Design*, 36(9):849–871, 2004.

[KYDH06] K-Y.Kim, D.G.Manley, and H.Yang. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design*, 38:1233–1250, 2006.

[KYK08] K-Y. Kim, H. Yang, and D-W. Kim. Mereotopological assembly joint information representation for collaborative product design. *Robotics and Computer-Integrated Manufacturing*, 24(6):744–754, 2008.

[Lee05] S.H. Lee. A CAD–CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. *Computer-Aided Design*, 37(9):941–955, 2005.

[LFB96] J. Lin, M. Fox, and T. Bilgic. A requirement ontology for engineering design. *Concurrent Engineering*, 4(3):279–291, 1996.

[LL70] D. Lewis and S. Lewis. Holes. *Australasian Journal of Philosophy*, 48(2):206–212, 1970.

[LLCN15] J. Liu, X. Liu, Y. Cheng, and Z. Ni. An approach to mapping machining feature to manufacturing feature volume based on geometric reasoning for process planning. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2015.

[Lou98] M.J. Loux. *Metaphysics: A contemporary Introduction.* Cambridge University Press, 1998.

[Low06] J.E. Lowe. *The Four-Category Ontology. A Metaphysical Foundation of Natural Science.* Oxford University Press, 2006.

[LY15] L. Lei and M. Yongsheng. CAD/CAE associative features for cyclic fluid control effect modeling. *Computer-Aided Design and Applications*, 2015.

[Ma13] Y-S. Ma, editor. *Semantic modeling and interoperability in product and process engineering. A technology for engineering informatics.* Springer-Verlag London, 2013.

[MB05] C. Masolo and S. Borgo. Qualities in formal ontology. In *Foundational Aspects of Ontologies (FOnt 2005) Workshop at KI*, pages 2–16, 2005.

[MBG$^+$03] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb deliverable D18. Technical report, Laboratory for Applied Ontology ISTC-CNR, 2003.

[MBTJ07] Y-S. Ma, G.A. Britton, S.B. Tor, and L.Y. Jin. Associative assembly design features: Concept, implementation and application. *International Journal of Advanced Manufacturing Technology*, 32(5-6):434–444, 2007.

[MCT08] Y-S. Ma, G. Chen, and G. Thimm. Paradigm shift: Unified and associative feature-based concurrent and collaborative engineering. *Journal of Intelligent Manufacturing*, 19(6):625–641, 2008.

[Mea15] P. J. Meadows. Holes cannot be counted as immaterial objects. *Erkenntnis*, 80(4):841–852, 2015.

[MH69] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence*, pages 431–450, 1969.

[Mil07] K. Miller. Immaterial beings. *The Monist*, 90(3):349–371, 2007.

[ML99] E. Margolis and S. Laurence. *Concepts: Core Readings.* Mit Press, 1999.

[MS08] K. Munn and B. Smith, editors. *Applied Ontology: An Introduction.* Walter de Gruyter, 2008.

[MSS02] H.K. Miao, N. Sridharan, and J.J. Shah. CAD-CAM integration using machining features. *International Journal of Computer Integrated Manufacturing*, 15(4):296–318, 2002.

[MVB+04] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social roles and their descriptions. In D. Dubois, C. Welty, and M.A. Williams, editors, *Principles of Knowledge Representation and Reasoning*, pages 267–277. AAAI Press, 2004.

[MVH+04] D.L. McGuinness, F. Van Harmelen, et al. OWL: Web ontology language overview. *W3C recommendation*, 2004.

[NK07] E.A. Nasr and A.K. Kamrani. *Computer-Based Design and Manufacturing: An Information-Based Approach.* Springer Verlag Berlin Heidelberg, 2007.

[NR08] F. Noël and L. Roucoules. The PPO design model with respect to digital enterprise technologies among product life cycle. *International Journal of Computer Integrated Manufacturing*, 21(2):139–145, 2008.

[NSFPZ13] M.P. Nepal, S. Staub-French, R. Pottinger, and J. Zhang. Ontology-based feature modeling for construction information extraction from a building information model. *Journal of Computing in Civil Engineering*, 27(5):555–569, 2013.

[PB13] G. Pahl and W. Beitz. *Engineering Design: a Systematic Approach.* Springer Science & Business Media, 2013.

[PBCV11] G. Pawel, S. Borgo, M. Carrara, and P. Vermaas. Two ontology-driven formalisations of functions and their comparison. *Journal of Engineering Design*, 22(11-12):733–764, 2011.

[PDS05] L. Patil, D. Dutta, and R. Sriram. Ontology-based exchange of product data semantics. *Automation science and engineering, IEEE transactions*, 2(3):213–225, 2005.

[PH96] B. Poldermann and I. Horvath. Surface design based on parametrized surface features. In *Proc. Int. Symposium on Tools and Methods for Concurrent Engineering, Institute of Machine Design, Budapest*, pages 432–446, 1996.

[PH16] S. Pourtalebi and I. Horvath. Towards a methodology of system manifestation features-based pre-embodiment design. *Journal of Engineering Design*, pages 1–37, 2016.

[PT16] P. Pauwels and W. Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable IfcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

[PUPS$^+$16] C. Palmer, E.N. Urwin, J.M. Pinazo-Sánchez, F.S. Cid, E.P. Rodríguez, S. Pajkovska-Goceva, and R.I.M. Young. Reference ontologies to support the development of global production network systems. *Computers in Industry*, 77:48–60, 2016.

[PZMY94] A. Pirotte, E. Zimanyi, D. Massart, and T. Yakusheva. Materialization: A powerful and ubiquitous abstraction pattern. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 630–641, 1994.

[QD04] X. Qian and D. Dutta. Feature-based design for heterogeneous objects. *Computer-Aided Design*, 36:1263–1278, 2004.

[RGB11] L. Ramos, A. Garcia, and J. Bateman. Ontology-based feature recognition and design rules checker system. In K. Baclawski, J. Bateman, A.G. Castro, C. Lange, and K. Viljanen, editors, *Proceedings of the Workshop Ontologies come of Age in the Semantic Web (OCAS 2011), 10th International Semantic Web Conference*, 2011.

[RHF$^+$06] S. Rachuri, Y-H. Han, S. Foufou, S.C. Feng, U. Roy, F. Wang, R.D. Sriram, and K.W. Lyons. A model for capturing product assembly information. *Journal of Computing and Information Science in Engineering*, 6(1):11–21, 2006.

[Rie03] M. Riebisch. Towards a more precise definition of feature models. In M. Riebisch, J.O. Coplien, and D. Streitferdt, editors, *Modelling Variability for Object-Oriented Product Lines*, pages 64–76. Norderstadt, 2003.

[RJB04] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language. The Reference Manual*. Pearson Higher Education, 2004.

[Ros90] J.R. Rossignac. Issues on feature-based editing and interrogation of solid models. *Computers & Graphics*, 14(2):149–172, 1990.

[RRB15] F. Romero, P. Rosado, and G.M. Bruscas. Application feature model for geometrical specification of assemblies. *Procedia Engineering*, 132:1128–1135, 2015.

[RS15] A. Rétfalvi and M. Stampfer. Aspects of clamping a workpiece over a through hole. In *IEEE 13th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 61–66. IEEE, 2015.

[Sal02] F. Sallustri. Mereotopology for product modelling. a new framework for product modelling based on logic. *Journal of Design Research*, 2, 2002.

[ŠAM10] M. Štorga, M.M. Andreasen, and D. Marjanovic. The design ontology: Foundation for the design knowledge exchange and management. *Journal of Engineering Design*, 21(4):427–454, 2010.

[San15] E.M. Sanfilippo. Towards an ontological formalization of technical product for design and manufacturing. In R. Cuel and R.I.M. Young, editors, *Formal Ontologies Meet Industry.*, volume 225, pages 75–87. Springer, 2015.

[SB16] E.M. Sanfilippo and S. Borgo. What are features? an ontology-based review of the literature. *Computer-Aided Design*, 80:9–18, 2016.

[SBF98] R. Studer, V.R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1):161–197, 1998.

[SBM14] E.M. Sanfilippo, S. Borgo, and C. Masolo. Events and activities: Is there any ontology behind BPMN? In P. Garbacz and O. Kutz, editors, *Formal Ontology in Information Systems.*

Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS), volume 267, pages 147–156. Springer Verlag Berlin Heidelberg, 2014.

[SBOW04] M.S. Shephard, M.W. Beall, R.M. O'bara, and B.E. Webster. Toward simulation-based design. *Finite Elements in Analysis and Design*, 40(12):1575–1598, 2004.

[SC10] B. Smith and W. Ceusters. Ontological realism: A methodology for coordinated evolution of scientific ontologies. *Applied ontology*, 5(3-4):139–188, 2010.

[SC15] B. Smith and W. Ceusters. Aboutness: Towards foundations for the information artifact ontology. In *Proceedings of the International Conference on Biomedical Ontology (ICBO) 2015*, 2015.

[Sch03] L. Schneider. How to build a foundational ontology. In *KI 2003: Advances in Artificial Intelligence*, pages 120–134. Springer, 2003.

[SCK$^+$05] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A.L. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome biology*, 6(5):1–15, 2005.

[Sea95] J.R. Searle. *The construction of social reality.* Simon and Schuster, 1995.

[SFFK$^+$03] S. Staub-French, M. Fischer, J. Kunz, K. Ishii, and B. Paulson. A feature ontology to support construction cost estimating. *AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(02):133–154, 2003.

[SFKS01] S. Szykman, S.J. Fenves, W. Keirouz, and S.B. Shooter. A foundation for interoperability in next-generation product development systems. *Computer-Aided Design*, 33:545–559, 2001.

[SFSW05] R. Sudarsan, S.J. Fenves, R.D. Sriram, and F. Wang. A product information modelling framework for product lifecycle management. *Computer-Aided Design*, 37(13):1399–1411, 2005.

[SG05] S. Subramani and Balan Gurumoorthy. Maintaining associativity between form feature models. *Computer-Aided Design*, 37(13):1319–1334, 2005.

[Sid01] T. Sider. *Four-Dimensionalism: An Ontology of Persistence and Time.* Oxford University Press, 2001.

[Sim87] P. Simons. *Parts. A Study in Ontology.* Clarendox Press Oxford, 1987.

[Sim96] H.A. Simon. *The Sciences of the Artificial.* MIT press, 1996.

[SLF+13] A. Shahwan, J-C. Léon, G. Foucault, M. Trlin, and O. Palombi. Qualitative behavioral reasoning from components' interfaces to components' functions for DMU adaption to FE analyses. *Computer-Aided Design*, 45(2):383–394, 2013.

[SM95] J.J. Shah and M. Mäntylä. *Parametric and Feature-Based CAD/CAM. Concepts, Techniques, Applications.* John Wiley and Sons, 1995.

[SM11] M. Sy and C. Mascle. Product design analysis based on life cycle features. *Journal of Engineering Design*, 22(6):387–406, 2011.

[SMBP16] E.M. Sanfilippo, C. Masolo, S. Borgo, and D. Porello. Features and components in product models. In R. Ferrario and W. Kuhn, editors, *Formal Ontology in Information Systems. Proceedings of the 9th International Conference (FOIS)*, volume 283, pages 227–240, 2016.

[Smi96] B. Smith. Mereotopology: A theory of parts and boundaries. *Data & Knowledge Engineering*, 20(3):287–303, 1996.

[SMN94] J.J. Shah, M. Mäntylä, and D.S. Nau. *Advances in feature based manufacturing.* Elsevier, 1994.

[SMP15] E.M. Sanfilippo, C. Masolo, and D. Porello. Design knowldege representation: An ontological perspective. In F.A. Lisi and S. Borgo, editors, *Proceedings of the 1st Workshop on Artificial Intelligence & Design (AIDE)*, volume 1473, pages 41–54. CEUR, 2015.

[SMR+13] B. Smith, T. Malyuta, R. Rudnicki, W. Mandrick, D. Salmen, P. Morosoff, D.K. Duff, J. Schoening, and K. Parent. IAO-intel: An ontology of information artifacts in the intelligence domain. In *STIDS*, 2013.

[SR88] Jami J. Shah and Mary T. Rogers. Functional Requirements and Conceptual Design of the Feature-based Modelling System. *Computer-Aided Engineering Journal*, 5(1):9–15, 1988.

[SRR16] L. Solano, F. Romero, and P. Rosado. An ontology for integrated machining and inspection process planning focusing on resource capabilities. *International Journal of Computer Integrated Manufacturing*, 29(1):1–15, 2016.

[SS13] S. Staab and R. Studer, editors. *Handbook on Ontologies.* Springer Science & Business Media, 2013.

[Sti90] G. Stiny. What is a design. *Environment and Planning B: Planning and Design*, 17(1):97–103, 1990.

[Sto98] O. Stock, editor. *Spatial and Temporal Reasoning.* Springer Science & Business Media, 1998.

[Str88] A. Stroll. *Surfaces.* University of Minnesota Press, 1988.

[SVHK93] O.W. Salomons, F. Van Houten, and H.J.J. Kals. Review of research in feature-based design. *Journal of Manufacturing Systems*, 12(2):113–132, 1993.

[SW01] B. Smith and C. Welty. Ontology: Towards a new synthesis. In B. Smith and C. Welty, editors, *Formal Ontology in Information Systems. Proceedings of the 2nd International Conference (FOIS)*, pages 3–9. ACM, 2001.

[SWS93] M. Schulte, C. Weber, and R. Stark. Functional features for design in mechanical engineering. *Computers in Industry*, 23(1):15–24, 1993.

[TCM13] S-H. Tang, G. Chen, and Y. Ma. Fundamental concepts of generic features. In Y-S. Ma, editor, *Semantic Modeling and Interoperability in Product and Process Engineering*, pages 89–115. Springer, 2013.

[Tho99] A.L. Thomasson. *Fiction and metaphysics.* Cambridge University Press, 1999.

[Tho14] A.L. Thomasson. Public artifacts, intentions, and norms. In M. Franseen, P. Kroes, T. Reydon, and P. Vermaas, editors, *Artefact Kinds. Ontology and the Human-Made World*, pages 45–62. Springer-Verlag Berlin Heidelberg, 2014.

[UE00] K.T Ulrich and S.D. Eppinger. *Product design and development.* MacGraw-Hill, 2000.

[UIY$^+$96] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama. Supporting conceptual design based on the Function-Behavior-State modeler. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 10(04):275–288, 1996.

[UM15] M. Uddin and Y-S. Ma. A feature-based engineering methodology for cyclic modeling and analysis processes in plastic product development. *Computer-Aided Design and Applications*, pages 1–12, 2015.

[Usm12] Z. Usman. *A Manufacturing Core Concepts Ontology to Support Knowledge Sharing.* PhD thesis, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, 2012.

[UY14] E.N. Urwin and R.I.M. Young. The reuse of machining knowledge to improve designer awareness through the configuration of knowledge libraries in plm. *International Journal of Production Research*, 52(2):595–615, 2014.

[UYC$^+$11] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case, and J. Harding. A manufacturing core concepts ontology for product lifecycle interoperability. In M. van Sindered and P. Johnson, editors, *Enterprise Interoperability. Proceedings for the Third International IFIP Working Conference.*, volume LNBIP 76, pages 5–18, 2011.

[UYC$^+$13] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case, and J. Harding. Towards a formal manufacturing reference ontology. *International Journal of Production Research*, 51(22):6553–6572, 2013.

[Var03] A.C. Varzi. Reasoning about space: The hole story. *Logic and Logical Philosophy*, 4:3–39, 2003.

[Var10] A.C. Varzi. On the boundary between material and formal ontology. *Interdisciplinary ontology*, 3:3–8, 2010.

[Var15] A. Varzi. Boundary. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Winter 2015 edition, 2015.

[VBM08] L. Vieu, S. Borgo, and C. Masolo. Artefacts and roles: Modelling strategies in a multiplicative ontology. In C. Eschenbach and M. Grüninger, editors, *Formal Ontology in Information Systems. Proceedings of the 5th International Conference (FOIS).*, volume 183, pages 121–134. IOS Press, 2008.

[VCBG13] P.E. Vermaas, M. Carrara, S. Borgo, and P. Garbacz. The design stance and its artefacts. *Synthese*, 190(6):1131–1152, 2013.

[VHB97] W. Van Holland and W.F. Bronsvoort. Assembly features and sequence planning. In M.J. Pratt, R.D. Sriram, and M. J. Wozny, editors, *Product Modeling for Computer Integrated Design and Manufacture*, pages 275–284. Springer, 1997.

[VHB00] W. Van Holland and W.F. Bronsvoort. Assembly features in modeling and planning. *Robotics and computer-integrated manufacturing*, 16(4):277–294, 2000.

[Weg96] P. Wegner. Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287, 1996.

[WGGM14] X. Wang, N. Guarino, G. Guizzardi, and J. Mylopoulos. Towards an ontology of software: a requirements engineering perspective. In P. Garbacz and O. Kutz, editors, *Formal Ontology in Information Systems. Proceedings of the 8th International Conference (FOIS)*, volume 267, pages 317–329, 2014.

[Whi04] D.E. Whitney. *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development.* Oxford University Press, 2004.

[Win91] L. Wingård. *Introducing form features in product models: a step towards CAD/CAM with engineering terminology.* PhD thesis, Dep. of Manufacturing Systems, Royal Institute of Technology, Stockholm, 1991.

[WP88] P.R. Wilson and M.J. Pratt. A taxonomy of features for solid modeling. In H.W. McLaughlin and J.L. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 125–136. North-Holland, 1988.

[WPY10] Q. Wang, W. Peng, and X. Yu. Ontology based geometry recognition for STEP. In *IEEE international Symposium on Industrial Electronics (ISIE 2010)*, pages 1686–1691. IEEE, 2010.

[WRP06] W.F.Bronsvoort, R.Bidarra, and P.J.Nyirenda. Developments in feature modelling. *Computer-Aided Design and Applications*, 3(5):655–664, 2006.

[WY14] Q. Wang and X. Yu. Ontology-based automatic feature recognition framework. *Computers in Industry*, 65(7):1041–1052, 2014.

[Xu09] X. Xu. *Integrating Advanced Computer-Aided Design, Manufacturing, and Numerical Control: Principles and Implementations.* Information Science Reference, Hershey New York, 2009.

[YB92] R.I.M. Young and R. Bell. Machine planning in a product model environment. *International Journal Of Production Research*, 30(11):2487–2513, 1992.

[YJ06] H. Yongtao and M. Jingying. A knowledge-based auto-reasoning methodology in hole-machining process planning. *Computers in Industry*, 57(4):297–304, 2006.

[YM16] Y. Yusuf and Y-S. Ma. Design of a simulation tool for steam assisted gravity drainage: Based on the concept of unified feature modeling scheme. In I. Horvath, J-P. Pernot, and Z. Rusak, editors, *Proceedings of TMCE 2016*, 2016.

[ZBLDE14] C. Zheng, M. Bricogne, J. Le Duigou, and B. Eynard. Survey on mechatronic engineering: A focus on design methods and product models. *Advanced Engineering Informatics*, 28(3):241–257, 2014.

[ZHX09] Y.F. Zhao, S. Habeeb, and X. Xu. Research into integrated design and manufacturing based on STEP. *International Journal of Advanced Manufacturing Technology*, 44:606–624, 2009.

[ZQH$^+$07] X. Zhou, Y. Qiu, G. Hua, H. Wang, and X. Ruan. A feasible approach to the integration of CAD and CAPP. *Computer-Aided Design*, 39(4):324–338, 2007.

[ZS04] X.F. Zha and R.D. Sriram. Feature-based component model for design of embedded systems. In *Optics East*, pages 226–237. International Society for Optics and Photonics, 2004.

[ZTT12] Y. Zheng, J.M. Taib, and M.M. Tap. Decomposition of interacting machining features based on the reasoning on the design features. *The International Journal of Advanced Manufacturing Technology*, 58(1-4):359–377, 2012.