Journal of Engineering Design
Vol. 22, Nos. 11–12, November–December 2011, 733–764

Taylor & Francis
Taylor & Francis Group

Two ontology-driven formalisations of functions and their comparison

Pawel Garbacz[a]*, Stefano Borgo[b], Massimiliano Carrara[c] and Pieter E. Vermaas[d]

[a]*Philosophy Department, John Paul II Catholic University of Lublin, Lublin, Poland;* [b]*Institute of Cognitive Sciences and Technologies (CNR) and KRDB (FUB), Trento, Italy;* [c]*Philosophy Department, University of Padua, Padova, Italy;* [d]*Philosophy Department, Delft University of Technology, Delft, The Netherlands*

(*Received 18 August 2010; final version received 13 June 2011*)

In this paper, we give formalisations of two engineering concepts of technical function and present in more general terms the project of supporting functional description translation by ontological analysis. The formalisations are given within the foundational DOLCE ontology and the concepts formalised are as follows: (1) the function as defined in the *Functional Representation* approach by Chandrasekaran and Josephson and (2) the function as defined in the *Functional Basis* approach by Stone and Wood. These two concepts represent two main ways of understanding functions in engineering: the first by means of the behaviour of artefacts, and the second by means of operations on flows as performed by artefacts. We analyse the similarities and differences between these concepts by means of the formalisations and show how the formalisations support the automated translation between functional descriptions based on these two concepts. In addition, we compare our strategy of formalising different engineering concepts of function within one foundational ontology with other strategies in the ontology-driven formalisation, such as defining a single formalised concept of function, either for replacing existing engineering concepts, or for use as a reference by which such existing concepts can be related. We compare these strategies and sketch the merits and shortcomings of our strategy.

Keywords: function; formal ontology; DOLCE; functional description translation; formalisation strategy

1. Introduction

There is a growing awareness in the engineering design community that proper understanding and efficient management of engineering knowledge are needed for developing information-based engineering techniques and for progressing toward 'the integrated product life-cycle'. Applied ontology provides theoretical tools and methodologies to study and clarify the conceptualisations presupposed in design processes (Kitamura *et al.* 2004, Borgo *et al.* 2009a), and in information structures for collecting, storing and retrieving engineering knowledge in general (Chandrasekaran

*Corresponding author. Email: garbacz@kul.pl

ISSN 0954-4828 print/ISSN 1466-1837 online

http://dx.doi.org/10.1080/09544828.2011.597376
http://www.tandfonline.com

et al. 1999, Ahmed *et al.* 2007, Witherell *et al.* 2007). The latter goal is described by Ahmed *et al.* (2007) in the following way:

> The motivation for developing an ontology includes knowledge sharing, and developing a standard engineering language. One item of particular interest is to provide a structured basis for navigating, browsing, and searching information through the hierarchical descriptions of the ontology. This is especially useful when designers are not aware of what kind of information is available or have difficulty in forming suitable queries. Designers can retrieve the documents by submitting natural language queries or navigating the ontology space. As a shared vocabulary, an ontology provides unified semantics of indexing and searching information, leading to improved accuracy for retrieval since the designers do not need to select the precise terms. (Ahmed *et al.* 2007, p. 132)

In this paper, we aim at contributing to this goal by developing tools for the sharing of functional descriptions in engineering. The paper extends previous efforts in using the methods of applied ontology to represent functional knowledge in engineering (Borgo *et al.* 2009a, 2009c, 2010a, 2010b). Up to this point, we have developed two formal theories, one for the notion of function as understood by the *Functional Representation* approach formulated by Chandrasekaran (1994) and Chandrasekaran and Josephson (2000), and one for the function as understood by the *Functional Basis* approach as defined by Stone and Wood (2000).[1] These two approaches have been identified in Chandrasekaran (2005) as giving the two main meanings of function in engineering. We have used for both formalisations the same upper-level formal ontology DOLCE (the acronym for *Descriptive Ontology for Linguistic and Cognitive Engineering*[2]), by which we obtain two important results for the goal of sharing functional descriptions:

(1) a conceptually homogeneous analysis of the conceptual structure of each of the two meanings, and
(2) a uniform framework for a direct comparison of these meanings.

These results are, especially, of relevance in the field of design methodology, where the envisaged exchange of engineering knowledge is substantially hampered by the introduction of various meanings for the term 'function'. These meanings are listed in the introductions of special journal issues on functions (Chakrabarti and Blessing 1996, Chittaro and Kumar 1998, Stone and Chakrabarti 2005) and brought together in surveys dedicated to the topic (Erden *et al.* 2008, van Eck 2009, Vermaas 2009). Individual design methodologies may advance single and stable meanings for function, as in, e.g. Umeda and Tomiyama (1995), Umeda *et al.* (1996) and Stone and Wood (2000), letting ambiguity surface only when engineering knowledge is exchanged across methodologies. Yet, even within a framework of a particular design methodology, the term in question may be ambiguous. Goel and collaborators (Goel 1991, 1992, Bhatta and Goel 1994, 1997) have given a succession of *three* different characterisations of function. In the work by Gero and collaborators (Gero 1990, Rosenman and Gero 1998, Gero and Kannengiesser 2004), the meaning of function has changed back and forth, as analysed in Vermaas and Dorst (2007). And some design methodologists accept that function can simultaneously have different meanings within a design method (Chakrabarti 1998, Chandrasekaran and Josephson 2000, Srinivasan and Chakrabarti 2009). Chandrasekaran and Josephson (2000, p. 170, Section 5.4), for example, identify a 'range of meanings for the term "function" in engineering science' formed by the device-centric meaning, the environment-centric meaning, and some mixtures thereof (see also Section 3.1). If this variety of approaches towards the meaning of function is representative of the status of functions in engineering, our two results may pave the way towards building tools for supporting engineers in sharing functional descriptions: comparisons of the functional concepts advanced by different approaches become feasible, as well as translations between functional descriptions generated within different approaches.

This paper makes one step in this direction: our goal here is to present and compare the formalisations of the concepts of function as advanced by the Functional Representation and Functional Basis approaches. This task is a crucial prerequisite to the aforementioned rationale

of applied ontology. Consider the following paradigm use-case scenario. Assume that we need to exchange information between two automated data systems: one built, either implicitly or explicitly, on the model of functions defined in the Functional Representation approach, and the other designed according to the principles of the Functional Basis approach. The reliable and smooth information flow between these systems requires that their database components should be matched semantically and not *only* on the basis of syntactic similarities between the names of tables and fields or on the basis of some vague intuitions by IT personnel. The stable and inter-subjectively accessible principles that explain the mapping between these components may be derived from the ontological analysis of the engineering models, which shaped the architecture of those components. Our paper is to provide such analysis by showing the differences in the conceptualisations between these models. The results reported below attest that these models differ to a larger degree than initially might seem.

Our strategy towards realising the goal of sharing functional descriptions in engineering presupposes that the status of functions in engineering is indeed one of ambiguity, i.e. that engineers indeed use this term with different meanings. By this presupposition, we take distance from the strategy that is more commonly adopted in applied ontology, namely, to resolve conceptual ambiguities by introducing (through formalisation) a *single* precise concept.

In Section 2, we survey related work on the formalisation of function, and contrast our strategy with the more common strategy of arriving at one formalised concept of function. In Section 3, we limit the scope of our strategy to the formalisation of functions as defined in the Functional Representation and Functional Basis approaches and briefly introduce these approaches. In Section 4, we describe DOLCE, the foundational ontology by means of which we express our formalisations. In Section 5, we present the main line of the formalisation of functions in the Functional Representation approach,[3] and in Section 6, we give it for functions in the Functional Basis approach. Section 7 contains the comparison of these two models as interpreted with our formalisations. In the next Section 8, we implement the results of this comparison by providing an example of automated information exchange between two information systems built according to the two approaches. We end with a discussion of the achieved results and with drawing some tentative conclusions.

2. Related work in ontology-driven formalisations of functions

The formalisation of technical functions in applied ontology or knowledge representation is generally seen as furthering a preformal understanding of this concept. Functions are used ubiquitously in engineering, but despite their central importance, the concept is, as noted in the introduction, associated with various and, sometimes, conflicting meanings. Work on the formalisation of technical functions in applied ontology is meant to resolve this ambiguity and is currently being carried out in different ways. In principle, two choices need to be made before this type of research commences: one concerning the meaning of the concept of function that is formalised and the other concerning the background ontology to be used in this formalisation. For the first choice, one needs to reach a normative assessment of the current meanings attached to the concept of function in engineering. Because these current meanings are ambiguous and imprecise, one can opt for an approach of introducing a new but clear and formalised concept of function that *ipso facto* has a meaning different from the meanings used by engineers. A new concept may replace existing meanings in engineering, or act as a reference against which the existing engineering meanings are analysed, related to one another, and eventually translated into one another. Arp and Smith (2008) have recently taken this approach and made an initial proposal for a new notion of technical functions. Alternatively, one can opt for directly formalising the existing ways in which engineers use functions without the help of auxiliary notions. This is the approach taken

by the group of Mizoguchi and Kitamura (Kitamura *et al.* 2002, 2004, Kitamura and Mizoguchi 2003, 2004, 2006). A third approach to engineering has been adopted within the GFO initiative by Burek *et al.* (2009), and aims primarily at providing a structure for the *representation* of functions.

The second choice is a choice for an ontology that provides the proper representation language and the conceptual framework in which to embed the concept(s) of function. Eventually, this choice is a choice between a light-weight '*terminological*' *ontology*, consisting of a clear and well-related standard engineering terminology, and a *foundational ontology*, formulated in terms of general, axiomatised and philosophically motivated concepts Borgo (2007). All the formal approaches we have listed above carry out their proposals adopting a foundational ontology, although not one and the same ontology:[4]

- Arp and Smith use BFO: *B*asic *F*ormal *O*ntology (http://www.ifomis.org/bfo);
- Mizoguchi and Kitamura rely on YAMATO: *Y*et *A*nother *M*ore-*A*dvanced *T*op-level *O*ntology (http://www.ei.sanken.osaka-u.ac.jp/hozo/onto_library/upperOnto.htm);
- Herre, Burek and Loebe start from GFO: *G*eneral *F*ormal *O*ntology (http://www.onto-med.de/en/theories/gfo).

Without plunging into the details of their background ontology, we briefly characterise these alternative approaches, starting with the one by Mizoguchi, Kitamura and their collaborators. Beforehand, we should warn the reader that each of these approaches uses its own technical terminology, which is usually defined in the respective foundational ontology. Therefore, the terms they employ (e.g. 'continuant', 'process', etc.) are to be understood as defined there.

In a series of papers (Kitamura *et al.* 2002, 2004, Kitamura and Mizoguchi 2003, 2004, 2006), a modelling framework for the sharing of functional knowledge in engineering is developed. The current structure of this framework contains at least three main components (Kitamura and Mizoguchi 2006, pp. 244–246):

(1) a top-level ontology aimed at clarifying the meaning of concepts used in other parts of the framework, the most important of them being the concept of role;
(2) an ontology of technical devices (the so-called 'extended device ontology'), which represents such entities as devices, conduits, media, or behaviours;
(3) an ontology of functional concepts together with a model for ways of function achievements, where one can find the definition of engineering function, the taxonomy of functions, the principles of functional decomposition, etc.

The last two ontologies are rendered in the Function and Behaviour Representation Language (FBRL – see Sasajima *et al.* 1995). In our comparison, we focus mainly on the ontology of functional concepts.

The ontology of functional concepts defines a (base) *function* as a role that a certain behaviour plays in a function context, where *behaviour* is defined (in extended device ontology) as the change of an attribute value of an object from that at the input port of a device to that at the output port of the device. In short, functions are behaviours interpreted within teleological contexts (Kitamura and Mizoguchi 2003, pp. 155–156). The following quote succinctly characterises the difference between behaviours and functions:

> In comparison with behavior, function is related to the intention of a designer or a user (i.e. it has a teleological interpretation) and hence is context-dependent. A behavior can implement different functions according to the context. For example, a heat exchanger can be used as a heater or a radiator. The behavior is the same in any context, that is, a heat flow from the warmer fluid to the colder one […]. The functions of the heater and the radiator can be 'to give heat' and 'to remove heat', respectively. (Kitamura and Mizoguchi 2006, p. 240)

The functional concept ontology gathers about 220 base functions in four subsumption hierarchies. These functions are used, among other things, to set up functional decomposition structures

for technical artefacts. For capturing functional decomposition a distinction is made between functions and *ways of function achievements*. If a designer decomposes a given (overall) function into a series of sub-functions, then the general knowledge about this series, including temporal and causal relations between its subfunctions and the physical principles that support the decomposition, is called the *way of function achievement*. This distinction is seen by Kitamura and Mizoguchi as a useful tool to clarify a number of concealed confusions in engineering design:

> The concept of way of achievement enables us to detach 'how to achieve' (way) from 'what is achieved' (function). For example, 'to weld steel' is not just a function, but function with a way in which the steel is melted. It should be decomposed into the 'connecting function' and 'fusion way'. (Kitamura and Mizoguchi 2003, p. 157)

Besides base functions, Kitamura and Mizoguchi introduce what they call *meta-functions*:

> Moreover, in addition to base-functions, we identified *meta-functions*. In comparison to a base function which is a role of behavior (and device) for an operand, a meta-function is a collaborative role played by a *function* for *another function* such as ToDrive and ToProvide. (Kitamura and Mizoguchi 2006, p. 246)

The model of functions by Herre, Burek and Loebe is a formal foundational ontology, called 'the Ontology of Functions' (OF) (Burek *et al.* 2009), and is presented as part of GFO. Interestingly enough, the concept of function and the related concepts are not considered here as domain terms, but instead as top-level concepts applicable in various domains. GFO provides taxonomies of entities and relations, including universals, abstract and concrete individuals, time and space regions, and facts. The OF system represents the structure of a function by a quadruple that consists of

(1) a set of the functions labels, informally describing 'to do something';
(2) a set of the functions requirements that are necessary to be present if the function is to be realised;
(3) a goal of the function, which is an intentional entity established for some reason (e.g. desired) by an agent referring to a chunk of reality, which is (to be) affected by the function;
(4) a functional item of the function that indicates the role of entities executing a realisation of this function.

The model of engineering functions at stake is broad enough to make room for both dynamic and static functions, and for general and particular/specific functions.

The OF ontology defines also four types of relations that may relate engineering functions with one another:

(1) instantiation;
(2) subsumption;
(3) parthood;
(4) realisation.

Thus, one can say that one function is an instance/part of another or that one function subsumes other functions or that one function realises another function.

Arp and Smith (2008) define a *function* as a realisable-dependent continuant such that

(1) it has a bearer, i.e. the function depends on a certain (independent) continuant, which is usually mentioned as the complement of the description of the form 'function of',
(2) its type has usually realisations such that
    (a) they are processes in which that bearer participates,
    (b) they exist in virtue of the bearer's physical make-up,
    (c) this make-up is had by the bearer because of how the bearer came into existence.

Clause (2)c of this definition makes room for the distinction between two subtypes of functions: artifactual and biological ones. The former are related to the cases in which the bearer's physical

make-up is the result of design and intentional production. The latter are related to the cases where the bearer is part of an organism and its make-up is the result of 'the coordinated expressions of that organism's structural genes'. Arp and Smith define also the notion of role in such a way that it is (extensionally) disjoint with the notion of function:

> In contrast to function, *role* is a realizable entity whose manifestation brings about some result or end that is not typical of its bearer in virtue of the latter's physical structure. Rather, the role *is played* by an instance of the corresponding kind of continuant entity because this entity is in some special natural, social, or institutional set of circumstances […]. (Arp and Smith 2008, p. 47)[5]

The methodological assumptions that guide our approach are to be sharply contrasted to the assumptions made in the three approaches discussed above in this section. We do not *construct a new* notion of engineering function, as Herre, Burek and Loebe as well as Arp and Smith do, or draw upon a certain generalisation of engineering practice, as it is the case in the Mizoguchi–Kitamura system. Rather, we *reconstruct existing* notions and preserve as much as is possible of the content of the models being formalised. After all the use of these existing notions and the resulting ambiguous status of function in engineering may turn out to be beneficial to engineering (Vermaas 2009).

It is therefore *not* assumed in our approach that there exists one single notion of function that is suitable for all engineering contexts, although we also do not exclude this possibility. On the other hand, a presupposition that we do make is that all formalisations of functions constructed on this route are or can be embedded in the DOLCE ontology and be developed by means of the so-called directed middle-out method.[6] This is to say, each formalisation starts in the middle of the ontological hierarchy and goes downwards and upwards, but going upwards it heads towards the DOLCE top-level ontology.

3. Limiting the research scope

There exists a variety of meanings of 'function' in current engineering. The most extensive survey has been given by Erden *et al.* (2008), where 18 approaches towards giving functional descriptions are listed. A thorough way of carrying out our project would imply formalising all the plausible meanings advanced by these 18 approaches, an effort which may be simplified a bit by identifying similarities between the meanings. In order to make progress in a more sophisticated way, we focus on only two main approaches in understanding functions as identified in Chandrasekaran (2005). These approaches are the *Functional Representation* (FR) approach formulated by Chandrasekaran (1994) and Chandrasekaran and Josephson (2000), and the *Functional Basis* (FB) approach as advanced by Stone and Wood (2000).

This limiting of the research scope may be taken as somewhat arbitrary; a different selection from the 18 approaches may be justified as well.[7] Relying on Chandrasekaran (2005), we assume that these two meanings cover large part of the practices in which engineers reason about functions. Moreover, the two meanings that are selected are quite different. In the FR approach, functions are related to behaviours of artefacts, and then these behaviours are related to structural-physical descriptions of the artefacts. This approach thus grants a pivotal conceptual role to the term 'behaviour' and suggests a clear ontological ordering: technical artefacts have their physical structure; this structure, in interaction with a physical environment, gives rise to the artefacts' behaviour; and this behaviour then determine in some way the artefacts' functions. This pivotal role of behaviour is diminished in the FB approach. Functions of artefacts are modelled in the FB approach in terms of inputs and outputs, and through this modelling, the functions are directly related to the structural-physical descriptions of artefacts. In this second approach, there is thus no explicit role for the concept of behaviour in relating functions of artefacts with their structure. By these differences between the two approaches, we cover thus two quite distinct meanings of function.

3.1. *The Functional Representation approach*

Functions of artefacts in the FR approach advanced by Chandrasekaran and Josephson (2000) are defined in terms of behaviour. Chandrasekaran and Josephson isolate five engineering meanings of behaviour and two of function. The meanings of behaviour are characterised with the help of the primitive notion of *state variables* (to clarify the meanings, we also report the examples provided in Chandrasekaran and Josephson (2000)):

- behaviour as the value of some state variable of the artefact or a relation between such values at a particular instant.
  Example: the car rattled when the driver hit the curve.
- behaviour as the value of a property of the artefact or a relation between such values.
  Example: a lintel distributes the load to two sides.
- behaviour as the value of some state variable of the artefact over an interval of time.
  Example: the BHP[8] increased for a while, but then started decreasing.
- behaviour as the value of some output state variable of the artefact at a particular instant or over an interval.
  Example: the amplifier is behaving well; the output voltage is constant.
- behaviour as the values of all the described state variables of the artefact at a particular instant or over an interval.
  (No example given.)

Notice that for all five meanings, a behaviour of a technical artefact is partially objective and partially subjective. It has an objective aspect because it eventually depends on the properties or features of the artefact. Still, the very same behaviour has a subjective aspect: it depends on the designer(s) and, indirectly, on engineering practice for the choice of the state variables.[9]

The two meanings of function that Chandrasekaran and Josephson distinguish are called the *device-centric* and *environment-centric* meanings. A *device-centric function* of an artefact is a behaviour of the artefact that is selected and intended by some agent. It is a function that is described in terms of the properties and behaviours of the artefact only; an example of a device-centric function is 'making sound' in the case of an electric buzzer. An *environment-centric function* is in turn an effect or impact of this behaviour of the artefact on its environment provided this effect or impact is selected and intended by some agent. This kind of function is conceptually separated from the artefact that performs or is expected to perform this function; 'enabling a visitor to a house to inform the person inside the house that someone is at the door' is an environment-centric function of the buzzer.[10]

3.2. *The Functional Basis approach*

In the FB approach, the concept of behaviour has no role and the meaning of function is given in a more uniform way. The approach of Stone and Wood (2000) propounds to model the overall product functions, especially from the electro-mechanical and mechanical domain, as sets of connected elementary sub-functions. In line with the design methodology of Pahl *et al.* (2007), an overall product function is described by means of a verb-object form and graphically represented by a black-boxed operation on flows of materials,[11] energies and signals. Sub-functions are also described by verb-object forms but they correspond to well-defined basic operations on well-defined basic flows of materials, energies and signals. Basic operations and basic flows are limited in number and listed in shared libraries, given in Hirtz *et al.* (2003), which define the functional design space. The whole system is called a 'Functional Basis' (FB).[12]

Stone and Wood present their proposal as supporting the archiving, comparison, and communication of functional descriptions of existing artefacts, as well as the engineering designing of new artefacts. Archiving, comparison, and communication is assisted since the sub-functions into which overall product functions are decomposed, are described by means of the limited number of basic operations and basic flows. Designing of new artefacts is supported since the FB design methodology allows designers to make critical design decisions about the architecture of artefacts, that is, decisions about the decomposition of the overall product functions in sub-functions, in the early conceptual stage of designing at which only functional descriptions are considered. Moreover, this approach provides the means to find overall design solutions quickly since Stone and Wood require the sub-functions to be small and easily solvable in designing.

In the designing of new artefacts, the overall product function is given in terms of input/output flows and may initially be coarse-grained. Consider the following example: the overall product function of loosening/tightening screws, which is performed or ascribed to a power screwdriver, is defined by several input flows: electricity, human force, on/off signals and screws; and one output flow: looseness/tightness of screws. But when this overall product function is decomposed in terms of connected sub-functions with well-defined input and output flows, the overall product function can be modelled in more detail. The input flows are extended to include also relative rotation, and the output flows to include also heat and noise. The full-fledged functional decomposition is depicted in Figure 1.

Stone and Wood provide a glossary of the terms used in their account, which we resume here for later reference (Stone and Wood 2000, pp. 359–360):

- *Product function*: the general input/output relationship of an artefact having the purpose of performing an overall task, typically stated in the verb–object form.

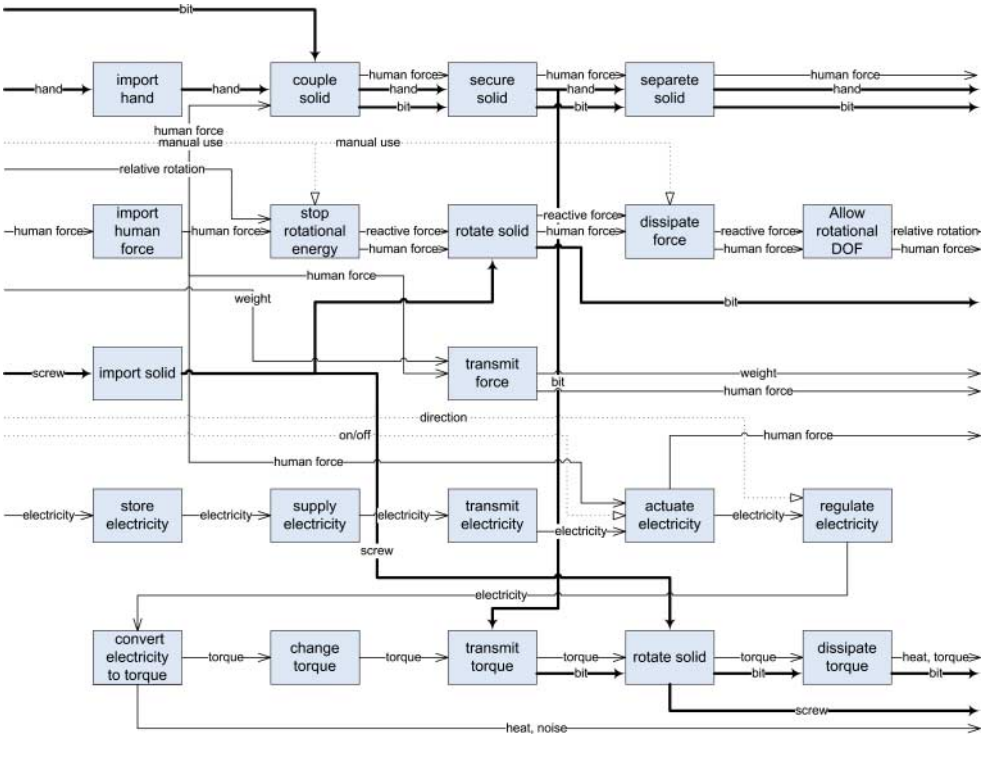

Figure 1. Functional decomposition of a screwdriver in FB (Stone and Wood 2000, p. 363).

- *Sub-function*: a description of a part of an artefact's overall task stated in the verb–object form. Sub-functions are decomposed from the product function and represent the more elementary tasks of the artefact.
- *Function*: a description of an operation to be performed by an artefact. It is expressed as the active verb of the sub-function.
- *Flow*: a change in material, energy or signal with respect to time. Expressed as the object of the sub-function, a flow is the recipient of the function's operation.

Note that the term 'function' *simpliciter* indicates a process (or, as they would say, an operation) and not the objects participating in this process (being operated upon). Formally, this definition has the awkward consequence that neither product functions nor sub-functions (as described above) are instances of functions in the FB terminology. More generally, this definition may create the wrong impression that the FB approach is merely one about operations, whereas it actually has a much broader scope and use in engineering.

4. DOLCE

The DOLCE ontology – the *Descriptive Ontology for Linguistic and Cognitive Engineering* – is part of the *WonderWeb* effort (see http://wonderweb.semanticweb.org). This effort aims to build a library of foundational ontology, each reflecting different ontological choices, i.e. perspectives on reality. Ontology in the *WonderWeb* is built with an extensive attention to a careful isolation of the ontological options and their formal relationships.

The DOLCE ontology is the most studied module of this library and is the one we choose as suitable for our goals. This means that we adopt DOLCE as a general guideline for our ontological analysis and use its structure as a tool to classify the engineering notions we formalise. This usage again introduces arbitrariness, which this time is unavoidable since a 'one and only', best ontology is currently not available, and may easily never come to be available. On the other hand, by formalising the FR and FB meanings of function within DOLCE, we also contribute to extend DOLCE to include concepts in the area of engineering. The outcome is that the ontology will be enriched with engineering notions, making it easier for the knowledge expert to further analyse, clarify and possibly improve even other areas in engineering design. Unfortunately, the foundational ontology like DOLCE are large and complex systems, so here we can provide just a minimal introduction. The interested reader can find in Masolo *et al.* (2002) the underlying motivations and a thorough discussion of technical aspects like the clarification of the ontology categories and the reasons to focus on *particulars* (individuals, tokens) as opposed to universals. Roughly speaking, a *universal* is an entity that is instantiated or concreted by other entities (like 'being human' and 'being an event'). A *particular*, an element of the class PARTICULAR, is an entity that is not instantiated by other entities (like the Eiffel Tower in Paris or Barack Obama). For example, a car you saw on the street is a particular as opposed to the general notion of a car, the latter is instantiated by a number of specific entities among which one is the car you saw. Particulars comprise physical entities, abstract entities, events and even qualities as we will see below.

The DOLCE ontology provides a flexible framework for the needs of engineering design: it adopts the distinction between objects like products and events like operations; it includes a differentiation among individual qualities (e.g. the weight of a specific material item), quality types (weight, colour, and the like), quality spaces (spaces to classify weights, colours, etc.), and quality positions or *qualia* (informally, locations in quality spaces). These, together with measure spaces (where the quality positions get associated to a measure system and to numbers), are important to describe and compare artefacts. Indeed, among the motivations to use DOLCE, an

important element was its robustness, flexibility and 'commonsense' perspective which allows to capture in a natural way the views proper of engineering practice.

The DOLCE ontology category ENDURANT comprises objects (like a *hammer*) or amounts of matter (like *an amount of plastic*, say, the specific plastic forming my mobile phone case), while the category PERDURANT comprises events like *making a hole* or *playing a soccer game*, that is, things that happen in time. (Formally, the category ENDURANT is represented by the constant ED and we write ED(x) to mean 'x is an endurant'. Analogously, we use PD for perdurants.) The term 'object' is used in the ontology to capture a notion of unity as suggested by the partition of the class PHYSICAL ENDURANT (formally, PED) into classes AMOUNT OF MATTER (M), FEATURE (F) and PHYSICAL OBJECTS (POB). Both endurants and perdurants are associated with a bunch of individual qualities (elements of the category QUALITY (QT)). The exact list of qualities may depend on the entity: *shape* and *weight* are usually taken as qualities of physical endurants,[13] *duration* and *direction* as qualities of perdurants. An individual quality, e.g. the weight of the car you are driving, is a quality associated with *one and only one* entity; it can be understood as the particular way in which that entity instantiates the general property 'having weight'. For example, the endurant Hammer_#321 (an individual) has its own individual instantiation of property 'having weight', namely, the individual weight-quality of Hammer_#321 which is what we measure when we put that hammer on a scale (if we put another hammer, no matter how similar, we would measure *another* individual quality, even if the scale indicates exactly the same value).

The change of an endurant in time is explained by the ontology through the change of some of its individual qualities. For example, with the substitution of a component, Hammer_#321 may change its weight. This means that the individual weight-quality of this entity was first associated to a position p and later to a position q of a given weight-quality space. Note that p and q should not be considered weight measures like, say, 5 kg. They are elements of (positions in) a quality space whose primary role is to partition individual qualities in equivalent (or similar, depending on the space) entities before committing to numeric values and measure units. Thus, the same p may be associated to 5 kg in one measure space, to 11.1 lb. in another, and to *fairly heavy* in yet another space. Quality spaces are elements of the REGION category, a subcategory of ABSTRACT where all these types of entities are coherently classified.

DOLCE's taxonomic structure is pictured in Figure 2. Each node in the graph is a category of the ontology. A category is a subcategory of another if the latter occurs higher in the graph and

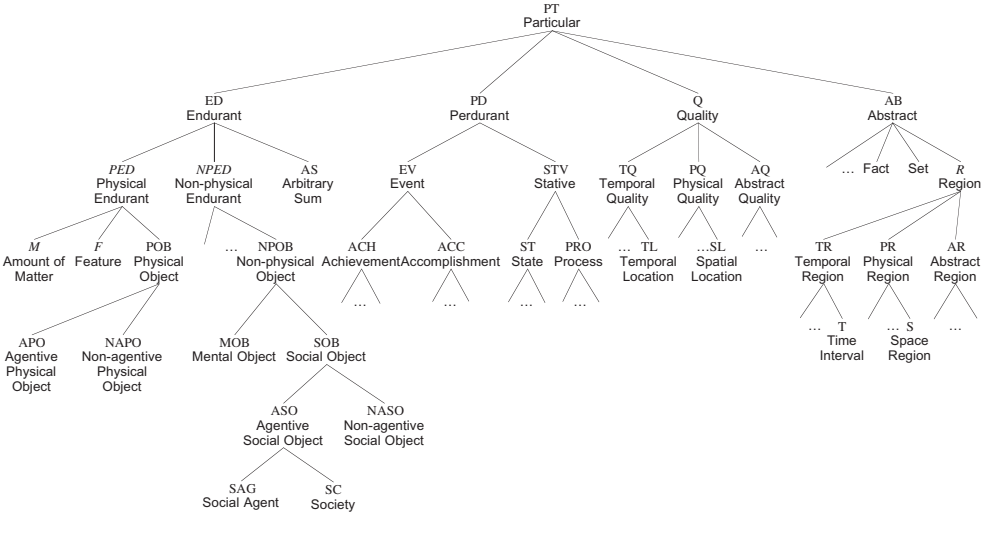

Figure 2. DOLCE taxonomy of entities.

there is an edge between the two. PARTICULAR is the top category. The class of subcategories of a given category forms a partition except where dots are inserted.

Another important relation in DOLCE is the *parthood* relation: 'x is part of y' (written: $\mathsf{P}(x, y)$), with its cognates the *proper part* (written: $\mathsf{PP}(x, y)$) and *overlap* relations (written: $\mathsf{O}(x, y)$). In DOLCE, the *parthood* relation applies to pairs of endurants (e.g. my arm is part of my body) as well as to pairs of perdurants (e.g. the voting of John Smith in 2008 US election is part of the 2008 US election event). For pairs of endurants, the relation of parthood is temporalised (i.e. with a third argument reserved to times) since an endurant may loose and gain parts throughout its existence (e.g. substituting a defective switch in my radio, the old switch is not part of my radio *after* the change while the new switch is).

5. Formalising Functional Representation functions

5.1. *Behaviours in Functional Representation*

Let a be a technical artefact, formally $\mathtt{TechArt}(a)$.[14] Let p be a perdurant, that is, $\mathsf{PD}(p)$. We take behaviour b of a in p to be *the specific way in which a occurs in p.* In this view, behaviour depends on the chosen a and p and is seen as a qualification of the participation relation. For instance, if a is a capacitor, then the way (manner) in which a occurs or exists in a given process of storing electric energy is a behaviour of precisely *this capacitor a*.[15]

Our definition of behaviour links behaviours with two categories of entities: endurants and perdurants. Formally, we take behaviour b to be an element of a new quality category $\mathtt{Beh}$, that is, a new subcategory of QT: it does not hold for a single endurant or perdurant, but for pairs of endurants and perdurants. In this way, it captures the special relationship between the artefact and the event in which it 'behaves'. Consequently, we are able to take into account inherent conceptual connections between behaviours and the entities to which we ascribe behaviours. We are also in a position to say that two different endurants behave differently in the same perdurant. For instance, if two capacitors (of different capacitances) in an electrical circuit participate in one process, we can say that they exhibit different behaviours (despite referring to the very same process) because they are associated to different entities.

To formalise this relationship, we introduce a ternary relation $\mathtt{Beh}(b, a, p)$, which reads 'b is the behaviour of the technical artefact a in perdurant p' and is taken as primitive in our theory. Then

$$\mathtt{Beh}(b, a, p) \longrightarrow \mathtt{TechArt}(a) \wedge \mathsf{PD}(p) \wedge \mathsf{QT}(b). \tag{1}$$

The unary predicate $\mathtt{Beh}$ will represent the class of all behaviours:

$$\mathtt{Beh}(b) \triangleq \exists a, p\, \mathtt{Beh}(b, a, p). \tag{2}$$

Looking at engineers' activity, we defined in Borgo *et al.* (2009a) different kinds of behaviour: actual, possible, impossible and generalised. While an 'actual behaviour' of an artefact is what it actually does during its life, the more general notion of 'possible behaviour' deals with what an artefact can possibly do. A pen may be destroyed before it happens to write, still the pen could have participated to a writing event, i.e. writing is part of its behaviour although not of its 'actual' behaviour. Furthermore, although a pen may not write due to a design flaw, still engineers, not aware of the flaw, talk about its writing behaviour. Consequently, we might be interested in perdurants that are only *believed* by rational agents to be possible (feasible), in particular, by engineers, but in which the artefacts at stake, designed or constructed, cannot do what is assumed of them. The union of actual, possible and physically impossible behaviours constitutes the class of generalised perdurants.

To incorporate dependence on agents' beliefs, needed to model Chandrasekaran and Josephson's notions of behaviour, two further notions of agent-related perdurants are introduced. Given a group of agents $\mathcal{G}$, we isolate the *$\mathcal{G}$-possible perdurants* using the predicate $\mathtt{PD}_{\mathcal{G}}(p)$ whose meaning is '$\mathcal{G}$ believes that p is a possible perdurant' and that we partially characterise in Borgo *et al.* (2009a). Then, a generalised engineering behaviour believed possible by $\mathcal{G}$, called a *$\mathcal{G}$-behaviour*, is defined by (symbol $\triangleq$ is used for definitions):

$$\mathtt{Beh}_{\mathcal{G}}(b,a,p) \triangleq \mathtt{Beh}(b,a,p) \wedge \mathtt{PD}_{\mathcal{G}}(p). \tag{3}$$

Note that, in accordance with the engineering understanding, the class of technical artefacts, TechArt, is here taken to be a subcategory of PHYSICAL OBJECT in the DOLCE ontology, that is

$$\mathtt{TechArt}(a) \longrightarrow \mathsf{POB}(a). \tag{4}$$

5.2. *Functions in Functional Representation*

Besides distinguishing different meanings of behaviour, Chandrasekaran and Josephson (2000) define two notions of function: *device-centric* function and *environment-centric*. Both of them presuppose a theoretical perspective in which functions are construed as intended behaviours. In this subsection, we show to what extent the ontological approach outlined above is suitable for grasping these concepts.

In order to characterise the two notions of functions, Chandrasekaran and Josephson start with the definition of the behavioural constraint. It is said in Chandrasekaran and Josephson (2000) that a *behavioural constraint* in a class of technical artefacts is any constraint on the behaviours of the elements of this class. As the examples given in Chandrasekaran and Josephson (2000) suggest, a behavioural constraint may be absolute, i.e. unconditional (e.g. that the value of the output voltage is greater than 5 V), or conditional (e.g. that if the input voltage is above 5 V, the output voltage is a sinusoid), so the most general form of behavioural constraint should be modelled as a pair of behaviours. If a behavioural constraint that consists of behaviours b_1 and b_2 is satisfied in the so-called behavioural environment r, we will write $\mathtt{SatCrBeh}(r,b_1,b_2)$. Thus it follows that

$$\mathtt{SatCrBeh}(r,b_1,b_2) \longrightarrow \mathtt{Beh}(b_1) \wedge \mathtt{Beh}(b_2). \tag{5}$$

5.2.1. *Device-centric functions*

The notion of device-centric function is described as follows:

> Let F be a class of behavioural constraints defined on, and satisfied by, an object D. If F is intended or desired by an agent A, then D has function F for A. (Chandrasekaran and Josephson 2000, p. 172)

To model device-centric functions, we lack only a notion of *behavioural constraint desired by* an agent. The predicate

$$\mathtt{DES}_{\mathcal{G}}(r,b_1,b_2),$$

meaning that 'the behavioural constraint (b_1,b_2) in environment r is desired by an agent $\mathcal{G}$', is introduced for this reason (in Borgo *et al.* (2009a) technical aspects of this predicate are discussed). Finally, we reach the sought definition:

$$\mathtt{DevFunc}_{\mathcal{G}}(a,b_1,b_2) \triangleq \mathtt{SatCrBeh}(a,b_1,b_2) \wedge \mathtt{DES}_{\mathcal{G}}(a,b_1,b_2). \tag{6}$$

'$\mathtt{DevFunc}_{\mathcal{G}}(a,b_1,b_2)$' is to be read as the behavioural constraint defined by behaviours b_1 and b_2 is a function of a that is intended by agent $\mathcal{G}$. It is the characteristic feature of device-centric functions

that their bearers, i.e. technical artefacts, coincide with their environments, i.e. configurations of entities in which these functions are satisfied.

Given formula (5), definition (6) implies that device-centric functions are pairs of behaviours or simply behaviours:

$$\texttt{DevFunc}_{\mathcal{G}}(a, b_1, b_2) \longrightarrow \texttt{Beh}(b_1) \wedge \texttt{Beh}(b_2). \tag{7}$$

5.2.2. *Environment-centric functions*

The environment-centric notion of function requires a further notion, that of *mode of deployment* for an artefact. It consists of (i) the structural relationships between the artefact and the objects in the environment and (ii) the actions in which the artefact and these objects are involved. Since these aspects are captured by perdurants, a mode of deployment for an artefact a in an environment r is simply any generalised perdurant p such that there exists artefact a_1 in the environment, i.e. $\mathsf{P}(a_1, r)$ and $a \neq a_1$ with both a and a_1 participating in p, formally:

$$\begin{aligned}\texttt{MD}(p, a, r) \triangleq\ & \texttt{TechArt}(a) \wedge \texttt{BehEnv}(r) \wedge \mathsf{P}(a, r) \\ & \wedge \exists a_1(\mathsf{P}(a_1, r) \wedge a \neq a_1 \wedge \texttt{PC}_{\texttt{WH}}(a, p) \wedge \texttt{PC}_{\texttt{WH}}(a_1, p)),\end{aligned} \tag{8}$$

where $\texttt{PC}_{\texttt{WH}}(a, p)$ means that a participates in the whole perdurant p. Then a *feasible mode of deployment*, FMD, is an (engineering) possible perdurant which is a mode of deployment. From Chandrasekaran and Josephson (2000):

> Let F be a class of behavioural constraints that an agent, say A, desires or intends to be satisfied in some [world] W. Let D be an object introduced into W, in a mode of deployment M(D, W). If D causes F to be satisfied in W, we say that D has, or performs, the function F in W. (Chandrasekaran and Josephson 2000, p. 171)

To model this, we need to fix a causal relation CS where $\texttt{CS}(p_1, p_2) \equiv$ 'perdurant p_1 causes perdurant p_2' and specialise it to SatCrBeh over r, b_1, b_2 as follows:

$$\begin{aligned}\texttt{CS}_{MD}(r, p, b_1, b_2) \triangleq\ & \texttt{SatCrBeh}(r, b_1, b_2) \wedge \forall p_1, p_2, a_1, a_2(\texttt{Beh}(b_1, a_1, p_1) \\ & \wedge \texttt{Beh}(b_2, a_2, p_2)) \longrightarrow (\texttt{CS}(p, p_1) \wedge \texttt{CS}(p, p_2)).\end{aligned} \tag{9}$$

'$\texttt{CS}_{MD}(r, p, b_1, b_2)$' means that the behavioural constraint defined by behaviours b_1 and b_2 is caused to be satisfied in environment r by perdurant p.

We can finally formalise the environment-centric function of Chandrasekaran and Josephson as follows:

$$\begin{aligned}\texttt{EnvFunc}(b_1, b_2, a, r, p) \triangleq\ & \texttt{CS}_{MD}(r, p, b_1, b_2) \\ & \wedge \texttt{FMD}(p, a, r)\ \wedge \exists \mathcal{G}\, \texttt{DES}_{\mathcal{G}}(r, b_1, b_2).\end{aligned} \tag{10}$$

'$\texttt{EnvFunc}(b_1, b_2, a, r, p)$' is to be read as the behavioural constraint defined by behaviours b_1 and b_2 is a function of an artefact a deployed into an environment r in a (feasible) mode of deployment p.

Again due to formula (5), definition (10) implies that environment-centric functions are pairs of behaviours or simply behaviours:

$$\texttt{EnvFunc}(b_1, b_2, a, r, p) \longrightarrow \texttt{Beh}(b_1) \wedge \texttt{Beh}(b_2). \tag{11}$$

Note also that from Equations (1), (7) and (11), we have

$$\begin{aligned}\texttt{DevFunc}_{\mathcal{G}}(a, b_1, b_2) &\longrightarrow \mathsf{QT}(b_1) \wedge \mathsf{QT}(b_2). \\ \texttt{EnvFunc}(b_1, b_2, a, r, p) &\longrightarrow \mathsf{QT}(b_1) \wedge \mathsf{QT}(b_2).\end{aligned} \tag{12}$$

The ontological framework sketched above has been used in Borgo *et al.* (2009a) to formalise the five meanings of behaviour and their examples, described in Chandrasekaran and Josephson (2000).

5.2.3. *Functions in Functional Representation – example*

Here, for reasons of space, we limit ourselves to formalising one example of a device centric function. Let Mini_12VCD be an electric buzzer that emits a sound (of a given frequency) when a switch to which the buzzer is connected (later on: SPST_1) is on. This means that

(1) Mini_12VCD participates in a (particular) perdurant $\texttt{make\_sound}_0$ that can be described as 'making a sound (of a given frequency and amplitude)' (informally one would say that the buzzer makes a certain sound).
(2) SPST_1 participates in a (particular) perdurant $\texttt{is\_on}_0$ that can be described as 'being on' (informally one would say that the switch is on).

Let $\texttt{b}_1$ represent the way that Mini_12VCD participates in $\texttt{make\_sound}_0$ – in this case $\texttt{b}_1$ points to the causal role of the buzzer due to which it generates the sound at stake. In our framework, we may encode this description simply by stating that

$$\texttt{Beh}(\texttt{Mini\_12VCD}, \texttt{make\_sound}_0, \texttt{b}_1).$$

Similarly, $\texttt{b}_0$ represents the way that SPST_1 participates in $\texttt{is\_on}_0$:

$$\texttt{Beh}(\texttt{SPST\_1}, \texttt{is\_on}_0, \texttt{b}_0).$$

Now the pair $\langle \texttt{b}_0, \texttt{b}_1 \rangle$ is a behavioural constraint out of which FR defines a function of Mini_12VCD.

If there is an agent, say John, who intends that $\langle \texttt{b}_0, \texttt{b}_1 \rangle$ is satisfied in this environment, then we can say that $\langle \texttt{b}_0, \texttt{b}_1 \rangle$ is a device-centric function of Mini_12VCD (for John), in short:

$$\texttt{DevFunc}_{\texttt{John}}(\texttt{Mini\_12VCD}, \texttt{b}_0, \texttt{b}_1).$$

6. Formalising Functional Basis functions

We now move to the Functional Basis approach which, as we saw earlier, relies on a very different framework of concepts.

6.1. *Design requirements*

A key concept in the formalisation of FB is the notion of *design requirements*. This notion is used by Pahl *et al.* (2007, Chapters 4.2 and 5) in a general sense comprising explicit and implicit requirements; it covers topics as different as demands and wishes of the user, economic feasibility, safety and performance requirements, and so on. However, when focusing on functional modelling, as in the work of Stone and Wood, the important requirements are just those that constrain the notion of engineering function broadly considered.

We write $\texttt{CustNeeds}(p, d, c)$ to indicate that product function p and the device d satisfy customer needs c (cf. Stone and Wood 2000).[16] Moreover, we write $\texttt{Req}(f, d, \mathit{fl}_i, \mathit{fl}_o, dr)$ to mean that the design requirements dr are satisfied by the engineering function f and device d with an input flow fl_i and an output flow fl_o.[17] In the functional design process, as envisioned in the

Functional Basis approach, the device is simply ignored. These two relationships are inter-linked by axioms (13) and (14) below.

Whenever the customer needs c are mandatory for the design requirements dr, we write $\texttt{Mand}(c, r)$. If $\texttt{Mand}(c, r)$ holds, then we can state that the satisfaction of the design requirements implies the satisfaction of the customer needs. Formally:

$$\texttt{Mand}(c, dr) \wedge \texttt{Req}(f, d, fl_i, fl_o, dr) \longrightarrow \texttt{CustNeeds}(f, d, c). \tag{13}$$

On the other hand, we have to ensure that customer needs are such only if they make sense from the FB perspective, that is, it is possible to find corresponding design requirements that can be satisfied:

$$\texttt{CustNeeds}(f, d, c) \longrightarrow \exists fl_i, fl_o, r\ (\texttt{Req}(f, d, fl_i, fl_o, dr) \wedge \texttt{Mand}(c, dr)). \tag{14}$$

6.1.1. *Design requirements and functions*

We say that a perdurant e is an engineering function with input flow fl_i and output flow fl_o if there is a design requirement dr and there is some device d which participates to the whole perdurant p such that $\texttt{Req}(p, d, fl_i, fl_o, dr)$:

$$\texttt{EngFun}(p, fl_i, fl_o) \triangleq \exists d, dr \texttt{Req}(p, d, fl_i, fl_o, dr). \tag{15}$$

From the ontological point of view, this definition neither implies nor presuppose that any particular specification of design requirements depends on the respective specification of functional description. At the same time, it neither implies nor presupposes the opposite: any particular specification of functional structure should depend on the respective specification of design requirements. All that definition (15) says is that for each engineering function, there exists a particular specification of design requirements and *vice versa*: for each specification of design requirements, there exists a corresponding engineering function (or functions).

An engineering function *simpliciter* is a perdurant for which there exist input and output flows that satisfy definition (15). Formally,

$$\texttt{EngFun}(p) \triangleq \exists fl_i, fl_o\ \texttt{EngFun}(p, fl_i, fl_o). \tag{16}$$

6.1.2. *Design requirements and flows*

FB partitions flows into three disjoint categories: material, energy and signal (cf. Pahl *et al.* 2007, pp. 29–30). This classification may be called ontological because it accounts for what flows *are*. In the perspective of functional descriptions, instead, we have seen that a (general) flow can participate to a function as an input or as an output flow. In other words, the function at stake provides the contextual perspective to classify flows as input or output flows. Again, we can define general flows via the notion of design requirements:

$$\texttt{InFlow}(fl, p) \triangleq \exists d, fl_o, dr\ \texttt{Req}(p, d, fl_i, fl_o, dr), \tag{17}$$

$$\texttt{OutFlow}(fl, p) \triangleq \exists d, fl_i, dr\ \texttt{Req}(p, d, fl_i, fl_o, dr), \tag{18}$$

where $\texttt{InFlow}(fl, p)$ and $\texttt{OutFlow}(fl, p)$ mean that fl is an input (output, respectively) flow for the engineering function p.

The correlation between the notions of input/output flow and that of function is now obvious:

$$\texttt{EngFun}(p, fl_i, fl_o) \equiv \texttt{InFlow}(fl_i, p) \wedge \texttt{OutFlow}(fl_o, p). \tag{19}$$

Since FB classifies flows in types, we enforce that any flow entity must belong to one of the allowed types, namely, material, energy or signal flow:

$$\texttt{Flow}(fl) \equiv (\texttt{MFlow}(fl) \vee \texttt{EFlow}(fl) \vee \texttt{SFlow}(fl)). \tag{20}$$

Of course, types must be distinct:

$$\texttt{MFlow}(fl) \longrightarrow \neg(\texttt{EFlow}(fl) \vee \texttt{SFlow}(fl)). \tag{21}$$

$$\texttt{EFlow}(fl) \longrightarrow \neg(\texttt{MFlow}(fl) \vee \texttt{SFlow}(fl)). \tag{22}$$

$$\texttt{SFlow}(fl) \longrightarrow \neg(\texttt{MFlow}(fl) \vee \texttt{EFlow}(fl)). \tag{23}$$

6.1.3. *Design requirements and devices*

As discussed, FB is not clear on its basic assumptions. This is the case even when the notion of device is used. Here we define a device as an object involved in a given context of design requirements. This assumption emphasises the centrality of the Req relationship in FB:

$$\texttt{Dev}(d) \triangleq \exists p, fl_i, fl_o, dr\, \texttt{Req}(p, d, fl_i, fl_o, dr). \tag{24}$$

One can then express the fact that the design requirements stipulate that a certain device performs a certain function:

$$\texttt{Perform}(d, p) \triangleq \exists fl_i, fl_o, dr\, \texttt{Req}(p, d, fl_i, fl_o, dr). \tag{25}$$

6.2. *Functional Basis and DOLCE*

6.2.1. *Ontological categorisation of FB functions*

As anticipated before, the engineering functions listed in FB are here interpreted as types whose instances are perdurants of certain kinds.

$$\texttt{EngFun}(p) \longrightarrow \mathsf{PD}(p). \tag{26}$$

The class of perdurants used by FB is not arbitrary and can be characterised at least to some extent. The proposed match between engineering functions and a subclass of perdurants allows us to provide a coherent and ontological framework for the three notions used in FB: function, sub-function and product function. For instance, given our assumption that an engineering function is the most basic in the ontological sense and is formalised by means of a class of perdurants, an FB sub-function is modelled as the subclass collecting all and only the perdurants that have as participants the flows requested in the (often informal) engineering description of the sub-function.

Given axiom (26) and Definition (15), it seems natural to constrain that an entity performing a function must participate to the whole perdurant:

$$\texttt{Perform}(p, d) \longrightarrow \texttt{PC}_{\texttt{WH}}(d, p). \tag{27}$$

6.2.2. *Ontological categorisation of FB flows*

The following three conditions make clear the ontological categories of FB *flow kinds*. They state that material flows are (a subclass of) endurants, e.g. a screw, that energy flows are (a subclass of) physical qualities of endurants, e.g. pressure, and that signal flows are (a subclass of) perdurants, e.g. visual signal:

$$\texttt{MFlow}(fl) \longrightarrow \mathsf{ED}(fl). \tag{28}$$

$$\texttt{EFlow}(fl) \longrightarrow \mathsf{PQ}(fl). \tag{29}$$

$$\texttt{SFlow}(fl) \longrightarrow \mathsf{PD}(fl). \tag{30}$$

Our full formalisation extends the above axioms and definitions with a number of additional constraints whose aim is to tie down the formal meaning of the concepts we use.

The above classification of flows needs some discussion. By taking energy flows as physical qualities, we may counter the engineering practice to see energy or energy flows as a separate ontological entity that exists or occurs independent of other entities. The mere fact that in FB engineers speak about energy flows that are distinct of the matter and signal flows illustrates this practice. Yet, also in engineering, it is acknowledged that energy flows in reality do not come separately. Pahl *et al.* (2007, p. 30) confirm that the conversion of electricity in a power plant is associated with a material conversion, although that latter conversion may not be visible in a nuclear power plant. We choose to see energy flows as qualities since this gives a clear ontological motivation of the flow distinction embraced by FB. If one insists on having energy flows as a separate category, this would lead to a revision of DOLCE or to the adoption of a different ontology.

For completeness, we point out an alternative option which is compatible with DOLCE: to model energy as an entity on a par with entities that are seen as 'amounts' like water and air. This view has some awkward consequences like making independent the change of energy and the change of water location when water is moving down a waterfall. Perhaps the uneasiness that some engineers show with this ontological classification is due to the constraint of looking at flows as objects: flows of energy are intrinsically associated by some to processes or events which have very different ontological properties. The whole issue is anyway subtle and might require some further discussion. Here we recognise the different positions while deciding to stay close to the view acknowledged by Pahl and Beitz. In FB, this position means to take energy as a primary flow, which presupposes a second carrier flow that has the purpose to transport the primary flow (Nagel *et al.* 2007).

Also, the claim that signal flows are perdurants should be explained. Clearly, not all perdurants are signal flows. Generally speaking, in engineering a signal flow refers to the presentation of data (geometric information, instruction list, etc.) and is typically provided by static perdurants or processes (a light is on, an increasing sound is emitted, a figure is shown, a flow chart is depicted in steps, etc.). The actual interpretation of the information (light on means that a specific temperature is reached, sound means the device is malfunctioning, figure means that the output port is on the back, etc.) is much more complicated and is not modelled by the FB approach.

6.2.3. *Ontological categorisation of FB devices*

In accordance with the engineering understanding, the class of devices, Dev, is here considered a subcategory of PHYSICAL OBJECT in the DOLCE ontology:

$$\texttt{Dev}(d) \longrightarrow \mathsf{POB}(d). \tag{31}$$

6.2.4. *Functions in Functional Basis – example*

Again, for reasons of space, we will provide only one example of our formalisation of FB. The example concerns an overall functional description of a power screwdriver (Figure 3). Its product function 'to loosen/tighten screws' is captured in our framework as a set of perdurants – see axiom (26). That is to say, we interpret the expression 'to loosen/tighten screws' as denoting the set of all perdurants such that each of them is a particular process in which certain particular screws are being either loosened or tightened.

Our categorisation of the flows involved in this case is presented in Table 1.[18]

In fact, we could provide a more adequate categorisation if we used the full strength of the DOLCE ontology. Namely, this ontology distinguishes various subcategories of endurants and perdurants (see Masolo *et al.* 2002), which allow us to capture our exemplary flows more precisely. For example, the product function 'to loosen/tighten screws' may be captured in DOLCE as a set of the so-called achievements. Table 2 gives the respective categorisation for the flows.

Figure 3. A functional description of the power screwdriver (from Stone and Wood 2000).

Table 1. An ontological categorisation of flows for the power screwdriver.

FB item	FB categorisation	Ontological categorisation
Electricity	Energy flow (EFlow)	Physical quality (PQ)
Human force	Energy flow (EFlow)	Physical quality (PQ)
Relative rotation	Energy flow (EFlow)	Physical quality (PQ)
Weight	Energy flow (EFlow)	Physical quality (PQ)
Hand	Material flow (MFlow)	Endurant (ED)
Bit	Material flow (MFlow)	Endurant (ED)
Screw	Material flow (MFlow)	Endurant (ED)
On/off	Signal flow (SFlow)	Perdurant (PD)
Manual use	Signal flow (SFlow)	Perdurant (PD)
Torque	Energy flow (EFlow)	Physical quality (PQ)
Heat	Energy flow (EFlow)	Physical quality (PQ)
Noise	Energy flow (EFlow)	Physical quality (PQ)
Looseness/tightness	Signal flow (SFlow)	Perdurant (PD)

Table 2. An extended ontological categorisation of flows for the power screwdriver.

FB item	Ontological categorisation
Hand	Agentive physical object
Bit	Non-agentive physical object
Screw	Non-agentive physical object
On/off	State
Manual use	Process
Looseness/tightness	State

7. Comparison

As one may expect, the two approaches, FR and FB, which we formalised above, turned out to lead to two very different ontological accounts.[19] Let us start our comparison with gathering the similarities. Both approaches aim at representing functions of technical artefacts, which in both cases turned out to be of the same ontological category, i.e. they are physical objects (cf. axioms (4) and (31) above). Moreover, the generality of the approaches suggests that the two classes of artefacts they address can be identified, corresponding to the adoption of the following axiom:

$$\texttt{TechArt}(x) \equiv \texttt{Dev}(x). \tag{32}$$

Prima facie, this might be seen as the single similarity between the two approaches. The reason is that these approaches are fundamentally different by relying on engineers' intentionality very differently. FR uses intentionality to identify functions locally since the analysis is carried out by considering what is a desired behaviour for a given device or within a given environment. In contrast, intentionality in FB serves as a foundation for a systematic view: it comes in to define what counts as basic function and what counts as flow, independently of specific situations.

First, note that our analysis of FR led to the conclusion that both device- and environment-centric functions eventually boil down to behaviours or relations between behaviours.[20] In our formalisation, this means individual qualities that characterise the way endurants occur in perdurants (see theorem (12) above). On the other hand, the FB approach presupposes that functions are perdurants themselves (see axiom (26) above). So the first difference at stake concerns the ontological *category* of technical functions. Since the category of physical qualities (QT) is disjoint in DOLCE from the category of perdurants (PD), this difference is ontologically significant. For instance, one cannot aim to simply map an FR function onto an FB counterpart (or *vice versa*) because the former is intrinsically a different object than the latter.

In order to succeed in relating the two approaches, one needs first a certain portion of ontological analysis to *reconstruct* a notion of one approach within the other system – see the example thereof below. In ontological terms, our reconstruction shows that both approaches are centred on perdurants. Nonetheless, FR builds upon a peculiar quality type relating to a perdurant and its participants, while FB discriminates two things: endurants and endurants' qualities on the one hand, and how a perdurant (the function) affects those, on the other. These ontological differences lead to conflicting implementations; it is then via an ontological analysis that we can hope to find a way to systematically translate information from one perspective into the other.

Secondly, the FR and FB approaches generate different structures of functional ascriptions. A functional ascription based on FR involves agents and, in the case of environment-centric functions, environments together with the respective modes of deployments. FB recognises explicitly neither of these factors. So in the former approach, a technical function is always a function *for* someone (if it is device-centric) and *in* something (if it is environment-centric), while the latter approach allows one to say that something (i.e. some perdurant) is a function *simplicter*. In other words, FR and FB differ in the canonical form of functional ascriptions:

(1) FR defines two forms:
   (a) x is a function (of artefact y) for agent z,
   (b) x is a function (of artefact y) for agent z in environment t in which y is deployed in mode v.
(2) FB defines one form:
   (a) x is a function (of artefact y).

In fact, the FB approach separates functions from technical artefacts that realise those functions. Consequently, neither representations of FB functions nor functional models mention representations of artefacts. Metaphorically speaking, FB functions are not modelled as functions *of* anything. On the other hand, both types of FR functions are explicitly modelled as functions of technical artefacts.

In order to better appreciate the ontological differences between these two methodologies, let us attempt to 'cross-exemplify' them, i.e. to model the FR example in the FB framework and *vice versa*.

It seems that the device-centric function of our Mini_12VCD, see Section 5.2.3, can be described in FB by means of the formula 'to convert electrical energy and signal into acoustic energy'. In our formalisation, this description will denote the set of all perdurants in which (a certain portion of) electrical energy and (a bit of) signal is converted into (a certain portion of) acoustic energy. Call these input flows, respectively, '$\texttt{electrical}_0$', '$\texttt{signal}_0$' and call the output flow '$\texttt{acoustic}_0$'. The term '$\texttt{electrical}_0 + \texttt{signal}_0$' will denote the mereological sum of these flows. $\texttt{make\_sound}_0$ is the perdurant considered in Section 5.2.3. According to axiom (29), electrical energy and acoustic energy are both flows of energy and are modelled as physical qualities.[21] In sum:

$$\texttt{EngFun}(\texttt{make\_sound}_0, \texttt{electrical}_0 + \texttt{signal}_0, \texttt{acoustic}_0).$$

$$\texttt{EFlow}(\texttt{electrical}_0).$$

$$\texttt{SFlow}(\texttt{signal}_0).$$

$$\texttt{EFlow}(\texttt{acoustic}_0).$$

$$\texttt{ExtFlow}(\texttt{electrical}_0 + \texttt{signal}_0).$$

$$\texttt{InFlow}(\texttt{electrical}_0, \texttt{make\_sound}_0).$$

$$\texttt{InFlow}(\texttt{signal}_0, \texttt{make\_sound}_0).$$

$$\texttt{OutFlow}(\texttt{acoustic}_0, \texttt{make\_sound}_0).$$

Note that 'to convert electrical energy into acoustic energy' mentions neither the agent who selects/intends this function nor its environment. As we have pointed out already, this functional representation has no explicit connection to the technical artefact itself, i.e. to Mini_12VCD.

This 'cross-exemplification' suggests a possible correlation between FR and FB models which we now exploit. Assume that the pair of behaviours z and z' are (device- or environment-centric) FR functions. If z represents the way in which an endurant x participates in a certain perdurant y, then y can be interpreted in FB as an instance of a certain function. In formal terms, our proposal is captured by the following:

$$\texttt{DevFunc}_{\mathcal{G}}(x, z, z') \wedge \texttt{Beh}(x, y, z + z') \rightarrow \texttt{EngFun}(y). \tag{33}$$

$$\texttt{EnvFunc}(z, z', x, x', y) \wedge \texttt{Beh}(x, y, z + z') \rightarrow \texttt{EngFun}(y). \tag{34}$$

Note that in all formulas from (12) to (34), endurant y may correspond to several FB functions depending on the flow(s) we identify or select in y.

Going in the opposite direction, our 'cross-exemplification' becomes much more demanding. This time we show how to map the FB function 'to loosen/tighten screws' onto the FR framework. Let PD600 be a power screwdriver whose function is to loosen and tighten screws. Let $\#\phi 20$ be a particular screw that can be loosened or tightened by means of the screwdriver. DOLCE taxonomy of perdurants allows us to say that each state in which $\#\phi 20$ is loosened and each state in which $\#\phi 20$ is tightened are perdurants. Let '$\texttt{loosened}_{\#\phi 20}$' denote a particular state in which $\#\phi 20$

is loosened and let '$\texttt{tightened}_{\#\phi 20}$' denote a particular state in which #ϕ20 is tightened. The FR framework allows us to pick up two behaviours of #ϕ20:

$$\texttt{Beh}(\#\phi 20, \texttt{loosened}_{\#\phi 20}, \texttt{b}_0),$$
$$\texttt{Beh}(\#\phi 20, \texttt{tightened}_{\#\phi 20}, \texttt{b}_1).$$

Behaviour $\texttt{b}_0$ characterises the way in which #ϕ20 participates in $\texttt{loosened}_{\#\phi 20}$. Similarly, for behaviour $\texttt{b}_1$. Note that these behaviours are behaviours of screw #ϕ20 and not of screwdriver PD600, so they are not behaviours of the bearer of the function at stake. The reason for focusing on them and not on a behaviour of PD600 is the fact that the FB function 'to loosen/tighten screws' is an environment-centric function as it is based on a specific effect that PD600 causes in its environment, namely that screw #ϕ20 is either loosened or tightened.

Having these two behaviours defined, we model two behavioural constraints:

(1) $\texttt{c}_0 = \langle \texttt{b}_0, \texttt{b}_1 \rangle$,
(2) $\texttt{c}_1 = \langle \texttt{b}_1, \texttt{b}_0 \rangle$.

Constraint $\texttt{c}_0$ identifies the process of tightening of #ϕ20, where #ϕ20 changes its state from $\texttt{loosened}_{\#\phi 20}$ to $\texttt{tightened}_{\#\phi 20}$. Constraint $\texttt{c}_1$ is related to the process of its loosening, where the reverse change occurs.

Now consider the environment of PD600, denoted $\texttt{env}_{\texttt{PD600}}$, which is a complex object (an endurant) comprising all the entities mentioned in Table 1, i.e.

- all FB flows that are endurants, namely:
  (1) hand,
  (2) bit,
  (3) PD600 itself,
  (4) screw #ϕ20,
- any endurant(s) that has the following physical qualities:
  (1) electricity,
  (2) relative rotation,
  (3) noise,
- any other endurant that participates in the on/off (signal).

For example, because of the flow of electricity, $\texttt{env}_{\texttt{PD600}}$ should contain a source of electrical energy or a live connection to an outside source. All other flows mentioned in Table 1 are included as qualities of one or more of the above entities. For instance, the flow of human force is a physical quality of the flow of hand mentioned above. The environment $\texttt{env}_{\texttt{PD600}}$ is the mereological sum of all these objects so that, for instance,

$$\mathsf{PP}(\texttt{PD600}, \texttt{env}_{\texttt{PD600}}),$$
$$\mathsf{PP}(\#\phi 20, \texttt{env}_{\texttt{PD600}}).$$

Since we deal with environment-centric functions, we also need to deploy PD600 in $\texttt{env}_{\texttt{PD600}}$ in such a way that the two behavioural constraints described above are satisfied in $\texttt{env}_{\texttt{PD600}}$:

$$\texttt{SatCrBeh}(\texttt{env}_{\texttt{PD600}}, \texttt{b}_0, \texttt{b}_1)$$
$$\texttt{SatCrBeh}(\texttt{env}_{\texttt{PD600}}, \texttt{b}_1, \texttt{b}_0)$$

To this end, we need to find perdurant $\texttt{deploy}_{\texttt{PD600}}$ such that it will represent a feasible mode of deployment of PD600 in $\texttt{env}_{\texttt{PD600}}$, i.e. such that

$$\texttt{FMD}(\texttt{deploy}_{\texttt{PD600}}, \texttt{PD600}, \texttt{env}_{\texttt{PD600}}).$$

Moreover, $\mathtt{deploy}_{\mathtt{PD600}}$ should cause the above two behavioral constraints to be satisfied in $\mathtt{env}_{\mathtt{PD600}}$, i.e.

$$\mathtt{CS}_{MD}(\mathtt{env}_{\mathtt{PD600}}, \mathtt{deploy}_{\mathtt{PD600}}, \mathtt{b}_0, \mathtt{b}_1),$$
$$\mathtt{CS}_{MD}(\mathtt{env}_{\mathtt{PD600}}, \mathtt{deploy}_{\mathtt{PD600}}, \mathtt{b}_1, \mathtt{b}_0),$$

Then when we state that both behavioural constraints are intended by some agent $\mathcal{G}$:

$$\mathtt{DES}_{\mathcal{G}}(\mathtt{env}_{\mathtt{PD600}}, \mathtt{b}_0, \mathtt{b}_1)$$
$$\mathtt{DES}_{\mathcal{G}}(\mathtt{env}_{\mathtt{PD600}}, \mathtt{b}_1, \mathtt{b}_0),$$

our ontological framework for FR becomes complete and we are now in a position to say that $\mathtt{c}_0$ and $\mathtt{c}_1$ (that is, pairs $\langle \mathtt{b}_0, \mathtt{b}_1 \rangle$ and $\langle \mathtt{b}_1, \mathtt{b}_0 \rangle$) are environment-centric functions:

$$\mathtt{EnvFunc}(\mathtt{b}_0, \mathtt{b}_1, \mathtt{PD600}, \mathtt{env}_{\mathtt{PD600}}, \mathtt{deploy}_{\mathtt{PD600}}),$$
$$\mathtt{EnvFunc}(\mathtt{b}_1, \mathtt{b}_0, \mathtt{PD600}, \mathtt{env}_{\mathtt{PD600}}, \mathtt{deploy}_{\mathtt{PD600}}).$$

Looking back at the general methodology behind these two cross-exemplifications, we can recast the overall idea motivated by ontological arguments, as follows: FB abstracts FR functions from information which is about 'how' to perform the desired behaviour. Focusing of objects' particular behaviours, FR functions represent not only what is achieved but also how it is achieved. This means that the FR functional representations are, as a rule, more specific than FB representations. Thus, if one goes from the former to the latter, one can obtain one FB function for each FR function. However, going in the opposite direction takes you from one FB function to a class of FR functions because the former does not take into account all the specifics required by any of the latter ones. One can say that an FB function identifies a *class* of FR functions by making explicit how the function can be physically performed. Indeed, physical constraints like the conservative laws can be fully expressed only within the FR viewpoint. The intentionality, which is only implicit in the FB function, forces the selection of the desired behaviours. These behaviours are then used to model the corresponding FR functions. This implies that integrating both approaches, the information system designer needs to deal with the phenomenon of (asymmetric) loss of information. Namely, we lose certain information, which concerns the specific aspects of the realisation of a given function, when we import an FR function to the information system that stores FB-based functional representations.

The ontologically-driven comparison we have exploited in these pages reveals one more benefit of using the ontological approach to engineering models. Having embedded both models in one foundational ontology we are in a position to reveal the substantial, deeply rooted differences between them and yet to provide solid grounds for information transfer across the two perspectives. The result of our comparison is different in quality and reliability from the research obtained using purely engineering methods as reported in Chandrasekaran (2005) as well as from other classification approaches like Kitamura *et al.* (2007).

8. Implementation

The previous section shows the conceptual relationships between the two engineering models or rather between their ontological presuppositions as exposed by our formalisations. In that section, the comparison has, however, been done 'manually', so to speak, so it cannot be directly applied to the architecture of information systems. In this section, we present how our formalisations may support the exchange of information between engineering design databases.

Table 3. From first-order logic ontology to OWL DL ontology.

Ontology	First-order category	OWL resource
FR	Beh (ternary)	FR:BehaviorRelReification owl:Class FR:behavior_endurant owl:ObjectProperty FR:behavior_perdurant owl:ObjectProperty
FR	Beh (unary)	FR:Behavior owl:Class
FR	$PD_{\mathcal{G}}$	No corresponding OWL entity
FR	TechArt	FR:TechnicalArtefact owl:Class
FR	SatCrBeh	FR:BehaviorSatisfactionRelReification owl:Class FR:behaviorSatisfaction_behavior owl:ObjectProperty FR:behaviorSatisfaction_environment owl:ObjectProperty
FR	$DES_{\mathcal{G}}$	FR:DesiredBehConstrRelReification owl:Class FR:behavioralConstraint_behavior owl:ObjectProperty FR:behavioralConstraint_agent owl:ObjectProperty
FR	MD	FR:ModeOfDeploymentRelReification owl:Class
FR	FMD	FR:FeasibleModeOfDeploymentRelReification owl:Class
FR	CS_{MD}	FR:CauseRelReification owl:Class FR:cause_environment owl:ObjectProperty FR:cause_perdurant owl:ObjectProperty
FR	$DevFunc_{\mathcal{G}}$	DeviceCentricFunction owl:Class
FR	EnvFunc	EnvCentricFunction owl:Class
FB	Mand	No corresponding OWL entity
FB	CustNeeds	No corresponding OWL entity
FB	Req	FB:RequirementRelReification owl:Class requirement_Device owl:ObjectProperty requirement_EngineeringFunction owl:ObjectProperty
FB	EngFun	FB:EngineeringFunction owl:Class
FB	InFlow	engineeringFunction_InputFlow owl:ObjectProperty
FB	OutFlow	engineeringFunction_OutputFlow owl:ObjectProperty
FB	MFlow	FB:MaterialFlow owl:Class
FB	EFlow	FB:EnergyFlow owl:Class
FB	SFlow	FB:SignalFlow owl:Class
FB	Flow	FB:Flow owl:Class
FB	ExtFlow	FB:FlowFB owl:Class
FB	Dev	FB:Device owl:Class
FB	Perform	FB: performs owl:ObjectProperty

To this end, we constructed two types of OWL DL ontologies, one for Functional Representation and the other for Functional Basis, as our two information silos. The OWL ontology simplify the formal theories sketched above in such a way that they become operational information artefacts. Table 3 shows the correspondence between the categories of the fully fledged formal theories and their OWL simplified counterparts. Note that both ontologies use the OWL Lite version of DOLCE (http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl).

To show how the information exchange between these systems might look like, we performed the following test.

(1) First, we populated the FB OWL ontology with the exemplary data available at http://designengineeringlab.org. For our purposes, we used the XML description of an air-purifier, which was generated by http://designengineeringlab.org. The translation process is implemented by a short JAVA algorithm within the Jena environment (cf. http://jena.sourceforge.net). The algorithm takes the XML description as an input data and creates the respective instances in the OWL FB ontology.
(2) Then, using the Jena API again, we implemented the mapping algorithm, whose pseudo-code is presented in Appendix 1 below. The mapping process automatically creates all the data needed for an FR model of engineering functions on the basis of the data available from the FB model. The resulting ontology imports the 'empty' OWL DL FR ontology and the OWL

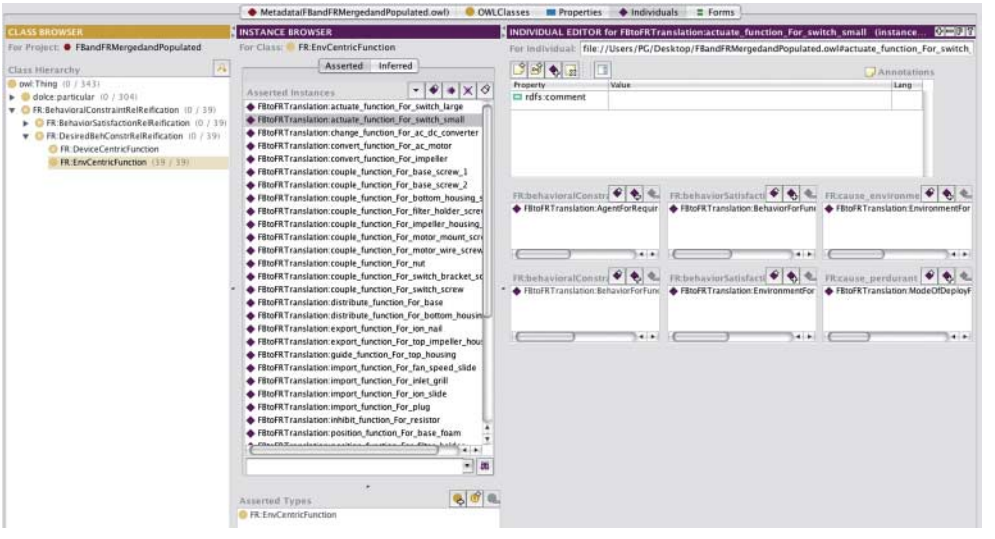

Figure 4. Common ontology in Protege.

DL FB ontology with instances and the mapping process creates the respective FR instances. Figure 4 shows the resulting ontology in the Protege editor.

The conceptual justification of the algorithm is derived from the 'manual' comparison given in the previous section, so it is eventually founded in our formal ontological theories. Its main assumption is that all basic FB functions may be translated as FR environment-centric functions. This assumption is based on the observation that the verb–noun representation is a black-box representation, that is, it is deliberately meant to abstract from the specific make up of the device concerned. This assumption does not exclude that some FB product functions need to be translated as device-centric FR functions.

Both ontologies and the code for the mapping algorithm are available at www.loa-cnr.it/FBFRmapping. The reader may invoke the whole process him/herself by running the jar-package application with the absolute path to the resulting ontology (to be created by the application) as the parameter (e.g. java -jar FBtoFRTransferFinal.jar/Users/John_Doe/FRPopulated.owl).

Figure 5 presents the structure of the whole system at stake in the form of a UML component diagram. We believe that this implementation exercise shows that ontological analysis may pave the way to the automatisation of the translation process between different models of technical devices. In each case, the respective models are to be ontologised. Then the resulting ontology needs to be rendered in some computer readable language. The ontological analysis should provide also with the details of the translation algorithm, so once we populate one 'computer-readable ontology' with the proper data, we will be able to automise the insertion process that will transport those data into the other model/ontology.

9. Discussion

We showed that the ontological analysis of the concept of function as used in engineering leads to identifying their essential elements and, subsequently, to their logical representation, which turns out to be quite revealing of the ties across the different approaches.

Yet it should be acknowledged that the above results focus primarily on the theoretical side of the function integration project we described in the introduction. Although we provided

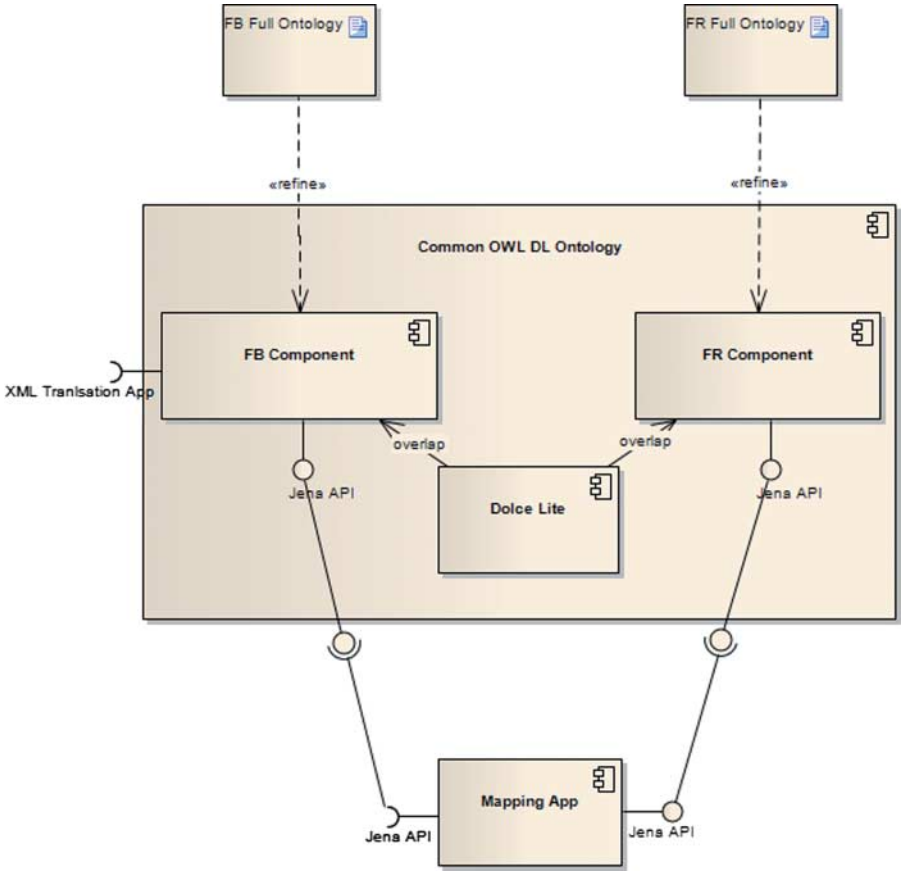

Figure 5. Implementation framework.

evidence of how our results lead to software implementation and of the potential benefits of such implementation, empirical proofs of the practical usefulness of the formalisations are still to be given. This validation may be established by experiments in more comprehensive implementations of our algorithm. Two preliminary issues raised by the formalisations can however already be addressed.

The first issue is whether we have built a sufficiently broad framework by covering only FR and FB functional descriptions. The second one is the relationship between our approach and the other ontological approaches like the one by Kitamura *et al.* (2004), Kitamura and Mizoguchi (2006) and Arp and Smith (2008).

Regarding the first issue, we relied on Chandrasekaran (2005) when taking the Functional Representation and the Functional Basis approaches as covering the main part of engineering. Yet it should be acknowledged that the term 'function' is used in engineering in many different ways. An analysis of the relevant meanings thereof may single out a different archetypical engineering approach which does not necessarily fit Chandrasekaran's choice. The number of archetypes can, for instance, be larger than two, as is exemplified also by the analysis of functions in Vermaas (2009, 2010). Conversely, it may be argued that we have been already too elaborated by formalising two concepts of functions. Chandrasekaran (2005) himself, for instance, makes a distinction between the FR and FB approaches, but also conjectures that the FB approach may eventually be taken as a special case of the FR approach. Similarly, in the survey in Erden *et al.* (2008), it is suggested

that all the engineering meanings identified may be partly integrated. And Vermaas (2010) argued that one can define one overall account of function that captures the three different archeypical meanings of functions Vermaas discerned. To some extent, the answer is that we are not in a position to determine whether the ontological analysis of functions will lead to a couple, to more than two, or eventually only one concept of function. An answer to this first issue is closely connected with different formalisation strategies for engineering functions.

The evaluation of the second issue, the relationship of our work with that of Mizoguchi and Kitamura, depends on how one considers the new formal concepts that ontological analyses may introduce into engineering. If these new concepts are again engineering meanings of functions, as is arguably the case for Mizoguchi and Kitamura, then comparing our work with that of Mizoguchi and Kitamura is required to address the issue just mentioned. If, however, these new concepts are not engineering meanings, as may be the case with the one defined by Arp and Smith, then the comparison should be carried out on different grounds. The motivation may rest on noting that in the work of Arp and Smith coherence between the concepts of biological function and technical function is central, which would mean that by comparing our formalisations with the one by Arp and Smith, the differences between FR functions, FB functions and biological functions can be determined. These differences may then be relevant to engineering fields in which biology plays a role, like biotechnology and biomimetic engineering. Let us notice in passing that recently Kitamura and Mizoguchi (2009) have claimed that their approach covers biological functions as well.

10. Conclusions

The term 'function' is associated in engineering with various and sometimes conflicting meanings, as the survey by Erden *et al.* (2008) of the 18 different engineering approaches to function shows. One consequence of this variety is that there might arise a certain type of communication problems when engineers exchange functional descriptions using different meanings. Ontological analysis can provide a solution to this communication problem: by laying down the different meanings within a single foundational ontology, the different concepts can be made explicit enabling automated functional reasoning using one meaning, and enabling automated translation of functional descriptions, using different meanings for the term. But it may be acknowledged that this solution still has a promissory character.

In this paper, we showed that such expectations are indeed satisfied to some relevant degree. We have followed the strategy of giving ontologically sound formalisations of two existing major concepts of function in engineering within one foundational ontology, and have provided a theoretically motivated as well as ontologically justified algorithm to practically translate functional knowledge from one representation approach to the other and *vice versa*. A first concluding remark is, therefore, that ontology can provide via this strategy a way to deal with this kind of integration and inter-operability problems. In ontology, other strategies are available as well. Rather than formalising existing concepts of function, one can introduce a new formalised concept of function, for relating the existing engineering concepts, or for replacing them. We suggested that these alternative strategies may become in conflict with engineering practices when it is beneficial for engineering to use the concept of function with more than one meaning. Yet, these strategies may still provide valuable ways to deal with the integration and inter-operability problems for functional knowledge.

The work presented in this paper following our strategy is not completed. Other concepts than the two advanced in the Functional Representation and the Functional Basis approaches may need to be formalised as well, and the presented formalisations may need to be developed into more detail on the basis of engineering literature on functions not considered in this paper. Moreover, the number and nature of practical illustrations of our formalisations show that the

translation of our ontological analyses to applications that are useful on the engineering work floor is still in an initial phase; our implementation has, for instance, been checked only for simple cases. A second concluding remark may thus be that the results presented in this paper are still at a development stage; their potential to solve the communication problem for functional descriptions in engineering is confirmed, but still has to be substantiated by practical applications.

Acknowledgements

This work has been developed while taking part in the Marie Curie 'EuJoint' project (IRSES 247503). Research by Pieter Vermaas for this paper is supported by the Netherlands Organisation of Scientific Research (NWO).

Notes

1. Both formalisations are presented here at a level of detail sufficient to our current goals of comparison and translation. For the unabridged versions the reader is referred to Borgo *et al.* (2009a, 2009c, 2010a, 2010b).
2. See Masolo *et al.* (2002).
3. Some parts of this paper originate from Borgo *et al.* (2009b).
4. Although foundational ontologies are meant to be stable systems, they are sometimes subject to changes. Typically this happens when the ontology is extended to include new concepts and relations and/or to include improvements in the formalisation. More rarely a foundational ontology undergoes fairly relevant structural changes, e.g. to simplify the use (and understanding) of the system or to embrace a more expressive structure. In these latter cases one might need to revise application systems that rely on the ontology. This however does not affect the overall ontological analysis inspired by the ontology, which is why ontological analyses, like the contribution we make in this paper, remain valid over time. We provide references to the ontologies' webpages so the reader will find the most current versions. Interestingly, a similar observation applies to the Functional Representation and Functional Basis approaches: these approaches can result in a variety of theories (differing in, say, the taxonomy of function, the taxonomy of flows, the relationships among flows, etc.) but since we mainly focus on their basic perspectives, our work remains of value even when switching from one version of these theories to another. Admittedly, some adaptation in our formal system might be needed, yet the overall system will remain in place.
5. In Carrara *et al.* (2011) we have argued against the adequacy of Arp and Smith's definition of function to capture some basic features of engineering functions. In fact, it excludes certain entities from the domain of functions although they are usually included in engineering models, and it includes certain objects as functions although they are rarely, if ever, included in engineering models. We return to this criticism in Section 3.1 after having introduced the distinction between *device-centric* and *environment-centric* functions in the *Functional Representation* system.
6. The term 'middle-out method' is used in the sense described in Uschold and Gruninger (1996).
7. For instance, three meanings are selected in Vermaas (2009, 2010) as archetypical to engineering. Two of these archetypical meanings correspond to the meanings considered in this paper.
8. BHP stands for Brake Horse Power and it is described as the amount of real horsepower going to the pump.
9. We discussed *partial objectivity* and *partial subjectivity* in Borgo *et al.* (2009a). This distinction is one that Chandrasekaran and Josephson made themselves. In particular, for the subjective aspect, they want to emphasise that a state variable of an artefact represents some feature or aspect of this artefact that might be relevant from a specific point of view.
10. Consider, again, Arp and Smith's definition of function. It seems that they would not acknowledge any of the *environment-centric functions* in the Functional Representation system, due to the fact that these functions exist not only because of the physical make-up of their bearers, but also, and mainly, because of the existence and make-up of other entities, which compose the environment concerned. So, their definition excludes certain entities from the domain of functions although they are usually included in engineering models.
11. The notion of material is construed here rather broadly as it comprises also such objects as screws, air, human beings, and their parts, e.g. human hands.
12. FB is the result of a reconciliation of two previous taxonomies: the NIST taxonomy (Szykman *et al.* 2000), and the older versions of FB developed in, for example (McAdams *et al.* 2000).
13. In DOLCE they are called physical qualities and are denoted by means of the PQ predicate.
14. The use of variables for our FR ontology is explained in Table A2 in the appendix.
15. Indeed, we begin by looking at capacitor instances (tokens) and do not address the behaviour of a *type* of capacitors.
16. In our formalisation of FB functions, we speak about devices rather than about artefacts. Stone and Wood themselves use the concepts of artefact, device and product side-by-side, without indicating whether there are differences between these concepts.
17. The use of variables for our FB ontology is explained in Table A4 in the appendix.
18. The classification of human force as physical quality might be challenged in DOLCE since human force, to be understood, requires to take time into consideration. In DOLCE this fact leads to think that human force should be seen as a quality of perdurants. However, in FB the term human force seems to be used quite broadly embracing

several views. We do not discuss these aspects in detail and, for the sake of the argument, take the simplest view here. This observation applies to other entries in this table like relative rotation and torque.

19. Since we compare in this section two different formal theories, we abstain here from the previously used variables and adopt the 'neutral' set of variables: x, y, z, etc. for the sake of comparison.
20. Formally speaking, they are *pairs* of behaviours.
21. Strictly speaking, they are subtypes of QT because they are types of physical qualities themselves.

References

Ahmed, S., Kim, S., and Wallace, K., 2007. A methodology for creating ontologies for engineering design. *The Journal of Computing Information Science and Engineering*, 7 (2), 132–140.

Arp, R. and Smith, B., 2008. Function, role, and disposition in basic formal ontology. *Proceedings of bio-ontologies workshop (ISMB 2008)*, Toronto, Canada, 5–48.

Bhatta, S. and Goel, A., 1994. Model-based discovery of physical principles from design experiences. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 8 (2), 113–123.

Bhatta, S. and Goel, A., 1997. Learning generic mechanisms for innovative design adaptation. *Journal of Learning Sciences*, 6 (4), 366–396.

Borgo, S., 2007. How formal ontology can help civil engineers. *In*: J. Teller, J. Lee, and C. Roussey, eds. *Ontologies for urban development*. Berlin: Springer, 37–45.

Borgo, S., *et al.*, 2009a. A formal ontological perspective on the behaviors and functions of technical artifacts. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23, 3–21.

Borgo, S., *et al.*, 2009b. Steps towards a formal ontology for engineering functions of technical artefacts. *Technical report Paper at the ICCME conference*, 2009, Salzburgh, Austria. Available from: http://importnet.salzburgresearch.at/ [Accessed 17 August 2010].

Borgo, S., *et al.*, 2009c. Towards an ontological representation of functional basis in DOLCE. *In*: M. Okada and B. Smith, eds. *Interdisciplinary ontology*, Vol. 2. Keio: Keio University Press, 3–16.

Borgo, S., *et al.*, 2010a. Formalizations of functions within the DOLCE ontology. *In*: F.M.I. Horváth and Z. Ruzák, eds. *Proceedings of the eighth international symposium on tools and methods of competitive engineering*. Delft: Delft University of Technology, 113–126.

Borgo, S., *et al.*, 2010b. An Ontological Interpretation of the Functional Basis Approach. Technical report Manuscript.

Burek, P., Herre, H., and Loebe, F., 2009. Ontological analysis of functional decomposition. *Proceeding of the 2009 conference on new trends in software methodologies, tools and techniques*. Amsterdam: IOS, 428–439.

Carrara, M., Garbacz, P. and Vermaas, P.E., 2011. If engineering function is a family resemblance concept: assessing three formalization strategies. *Applied Ontology*, 6 (2), 141–163.

Chakrabarti, A., 1998. Supporting two views of function in mechanical design. *15th national conference on artificial intelligence (AAAI'98)*, Wisconsin.

Chakrabarti, A. and Blessing, L., 1996. Representing functionality in design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10 (4), 251–253.

Chandrasekaran, B., 1994. Functional representation and causal processes. *Advances in Computers*, 38, 73–143.

Chandrasekaran, B., 2005. Representing function: relating functional representation and functional modeling research streams. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 19, 65–74.

Chandrasekaran, B. and Josephson, J., 2000. Function in device representation. *Engineering with Computers*, 16, 162–177.

Chandrasekaran, B., Josephson, J., and Benjamins, V., 1999. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14 (1), 20–26.

Chittaro, L. and Kumar, A., 1998. Reasoning about function and its applications to engineering. *Artificial Intelligence in Engineering*, 12 (4), 331–336.

van Eck, D., 2009. On relating functional modeling approaches: abstracting functional models from behavioral models. *eProceedings of the 17th international conference on engineering design*, 24–27 August 2009, Stanford, CA, 2.89–2.100.

Erden, M.S., *et al.*, 2008. A review of function modeling: approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22, 147–169.

Gero, J.S., 1990. Design prototypes: a knowledge representation schema for design. *AI Magazine*, 11 (4), 26–36.

Gero, J.S. and Kannengiesser, U., 2004. The situated function-behaviour-structure Framework. *Design Studies*, 25 (4), 373–391.

Goel, A., 1991. A model-based approach to case adaptation. *In*: *Proceedings of the 13th annual cognitive science conference*, August 1991, Chicago Hillsdale, NJ: Lawrence Erlbaum, 143–148.

Goel, A., 1992. Representation of design functions in experience-based design. *In*: D. Brown, M. Waldron, and H. Yoshikawa, eds. *Intelligent computer aided design*. Amsterdam, Netherlands: North-Holland, 283–308.

Hirtz, J., *et al.*, 2003. A functional basis for engineering design: reconciling and evolving previous efforts. *Research in Engineering Design*, 13 (2), 65–82.

Kitamura, Y. and Mizoguchi, R., 2003. Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications*, 24 (2), 153–166.

Kitamura, Y. and Mizoguchi, R., 2004. Ontology-based systematization of functional knowledge. *Journal of Engineering Design*, 15 (4), 327–351.

Kitamura, Y. and Mizoguchi, R., 2006. An ontological model of device function: industrial deployment and lessons learend. *Applied Ontology*, 1 (3), 237–262.

Kitamura, Y. and Mizoguchi, R., 2009. A device-oriented definition of functions of artifacts and its perspectives. *In*: U. Krohs and P. Kroes, eds. *Functions in biological and artificial worlds: comparative philosophical perspectives*. Cambridge, MA: MIT, 203–221.

Kitamura, Y., *et al.*, 2002. A functional concept ontology and its application to automatic identification of functional structures. *Advanced Engineering Informatics*, 16 (2), 145–163.

Kitamura, Y., *et al.*, 2004. Deployment of an ontological framework of functional design knowledge. *Advanced Engineering Informatics*, 18 (2), 115–127.

Kitamura, Y., Takafuji, S., and Mizoguchi, R., 2007. Towards a reference ontology for functional knowledge interoperability. *DETC2007-35373 Proceedings of the ASME 2007 IDETC/CIE conference*, 4–7 September, Las Vegas, Nevada, 111–120.

Masolo, C., *et al.*, 2002. *Wonderweb Deliverable d17: the Wonderweb library of foundational ontologies*. Trento: Laboratory for Applied Ontology, technical report.

McAdams, D.A., Stone, R.B., and Wood, K., 2000. Functional interdependence and product similarity based on customer needs. *Research in Engineering Design*, 11 (1), 1–19.

Nagel, R.L., *et al.*, 2007. A representation of carrier flows for functional design. *Proceedings of ICED'07*, 28–31 August, Paris, France.

Pahl, G., *et al.*, 2007. *Engineering design: a systematic approach*. 3rd ed. Berlin: Springer.

Rosenman, M.A. and Gero, J.S., 1998. Purpose and function in design: from the socio-cultural to the techno-physical. *Design Studies*, 19 (2), 161–186.

Sasajima, M., *et al.*, 1995. FBRL: a function and behavior representation language. *In: IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence*, Vol. 2. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1830–1836.

Srinivasan, V. and Chakrabarti, A., 2009. Sapphire: an approach to analysis and synthesis. *Proceedings of 17th international conference on engineering design (ICED'09)*, 24–27 August, Stanford University, CA, USA. Glasgow, UK: The Design Society, 417–428.

Stone, R.B. and Chakrabarti, A., 2005. Special issue: engineering applications of representations of function, part 1. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 19 (2), 63.

Stone, R.B. and Wood, K.L., 2000. Development of a functional basis for design. *Journal of Mechanical Design*, 122 (4), 359–370.

Szykman, S., Racz, J., and Sriram, R., 2000. The representation of function in computer-based design. *DETC99/DTM-8742 Proceedings of the 1999 ASME IDETC/CIE conferences*, September, Las Vegas, NV.

Umeda, Y. and Tomiyama, T., 1995. FBS modeling: modeling scheme of function for conceptual design. *Working Papers of the 9th Int. Workshop on Qualitative Reasoning About Physical Systems*, Amsterdam, 271–278.

Umeda, Y., *et al.*, 1996. Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10 (4), 275–288.

Uschold, M. and Gruninger, M., 1996. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11 (2), 93–136.

Vermaas, P.E., 2009. The flexible meaning of function in engineering. *eProceedings of the 17th international conference on engineering design*, 24–27 August 2009, Stanford, CA, 2.113–2.124.

Vermaas, P.E., 2010. Technical functions: towards accepting different engineering meanings with one overall account. *In*: I. Horváth, F. Mandorli, and Z. Ruzák, eds. *Proceedings of the eighth international symposium on tools and methods of competitive engineering*. Delft: Delft University of Technology, 183–194.

Vermaas, P.E. and Dorst, K., 2007. On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology. *Design Studies*, 28 (2), 133–157.

Witherell, P., Krishnamurty, S., and Grosse, I., 2007. Ontologies for supporting engineering design optimization. *The Journal of Computing Information Science and Engineering*, 7 (2), 141–150.

Appendix 1. Mapping algorithm

Algorithm A.1: MAPPING ALGORITHM: FB → FR(*FB ontology*, *FR ontology*)

```
req_List ← FINDALL( of FB:RequirementRelationReification)
for each req ∈ req_List
do {
    agent ← CREATEINST(FR:Agent)
    dev ← FINDSLOT(FB:requirement_Device, req)
    INSTANTIATE(FR:TechnicalArtefact, dev)
    func_List ← FINDSLOT(FB:requirement_EngineeringFunction, req)
    for each func ∈ func_List
    do {
        beh ← CREATEINST(FR:AgentRecognizedBehavior)
        FILLSLOT(FR:behavior_Agent, beh, agent)
        FILLSLOT(FR:behavior_endurant, beh, dev)
        FILLSLOT(FR:behavior_perdurant, beh, func)
        env ← CREATEINST(dolce:endurant)
        FILLSLOT(dolce:part-of, dev, env)
        deploy ← CREATEINST(FeasibleModeOfDeploymentRelReification)
        FILLSLOT(modeOfDeployment_environment, env)
        FILLSLOT(modeOfDeployment_TechnicalArtefact, dev)
        flow_List ← FINDSLOT(FB:engineeringFunction_Flow, func)
        for each flow ∈ flow_List
        do {
            if ISINST(flow, FB:EnergyFlow)
            then {
                end ← CREATEINST(dolce : endurant)
                FILLSLOT(dolce:inherent-in, end, flow)
                FILLSLOT(dolce:part-of, end, env)
            }
            if ISINST(flow, FB:SignalFlow)
            then {
                end ← CREATEINST(dolce : endurant)
                FILLSLOT(FR:whParticpate, end, flow)
                FILLSLOT(dolce:part-of, end, env)
            }
        }
        frfunc ← CREATEINST(FR:EnvCentricFunction)
        FILLSLOT(behaviorSatisfaction_behavior, frfunc, beh)
        FILLSLOT(behaviorSatisfaction_behavior, frfunc, beh)
        FILLSLOT(behavioralConstraint_agent, frfunc, agent)
        FILLSLOT(cause_perdurant, frfunc, deploy)
        FILLSLOT(cause_environment, frfunc, env)
    }
}
```

procedure FINDALL(*class*)
comment: finds all instance of class
return (*instanceList*)

procedure ISINST(*instance*, *class*)
comment: checks whether instance is an instance of class
return (*boolean*)

procedure CREATEINST(*class*)
comment: creates an instance of class
return (*instance*)

procedure INSTANTIATE(*class*, *instance*)
comment: makes instance an instance of class
return (*null*)

procedure FINDSLOT(*property*, *slotHolder*)
comment: find a resource in property slot of slotHolder
return (*resourceInSlot*)

procedure FILLSLOT(*property*, *slotHolder*, *resourceInSlot*)
comment: fill property slot of slotHolder with resourceInSlot
return (*null*)

Appendix 2. Technical nomenclature

A.1. FR *ontology*

Table A1. Predicates in FR.

Predicate name	Non-formal explanation
ED	Is an endurant
PD	Is a perdurant
PED	Is a physical endurant
POB	Is a physical object
P	Is part of
PP	Is a proper part of
PC_{WH}	Participates
TechArt	Is a technical artefact
Beh	Is a behaviour in an event
Cond	Is a condition for
CrBeh	Is a behavioural constraint in
SatCrBeh	Is a satisfied behavioural constraint in
DES	Behavioural constraint is desired in
BehEnv	Is an environment
CS	Causally brings about
DevFunc	Is a device-centric function
EnvFunc	Is an environment-centric function
MD	Is a mode of deployment
FMD	Is a feasible mode of deployment

Table A2. Variables in FR.

Variable	Range
$p, p_1, \ldots$	Perdurants
$a, a_1, \ldots$	Technical artefacts
$b, b_1, \ldots$	Behaviours
r	Behavioural environments
$\mathcal{G}$	Agents or agent communities
T	Time objects

A.2. FB *ontology*

Table A3. Predicates in FB.

Predicate name	Non-formal explanation
Req	Design requirements
CustNeeds	Customer needs
EngFun	Engineering function
BasicFunc	Basic function
ProdFunc	Product function
Dev	Is a technical device
InFlow	Is an input flow for
OutFlow	Is an output flow for
MFlow	Is a flow of matter
EFlow	Is a flow of energy
SFlow	Is a flow of signal
Perform	Performs
Mand	Is mandatory

Table A4. Variables in FB.

Variable	Range
c	Customer needs
dr	Design requirements
p	Perdurants
d	Devices
fl	Flows
$\mathrm{fl_i}$	Input flows
$\mathrm{fl_o}$	Output flows