INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITÉ DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

A TAXONOMY OF FUNCTIONS IN GELLISH ENGLISH

Andries van Renssen[1], Pieter E. Vermaas[2] and Sjoerd D. Zwart[2]
[1]Royal Dutch Shell
[2]Delft University of Technology

ABSTRACT
This paper sketches a possible route to overcome the problems of ambiguous use of the function notion in engineering. Our method is to implement functional descriptions in a formal language, viz. Gellish English, which is standardized structured English defined in STEPlib (based on ISO 10303-221 (AP221) and ISO 15926). Our main results are twofold. First, we provide a formal taxonomy in Gellish English of three different and important senses of the term function: the SE-function featuring in systems engineering colloquially characterized as the "actions carried out to achieve the system's objectives"; the KB-function, which is mainstream in knowledge base architectures as "being the performer of the activity," and the C-function in philosophy as "the capacity of the performing object." Second, we show how to prune philosophical accounts of functions down to manageable detail for practical engineering applications.

Keywords: Functions, knowledge bases, systems engineering, capacities, STEPlib, Gellish English

1 INTRODUCTION
Although functional descriptions are abundant in engineering, their meaning is often ambiguous [1], [2], [3]. This ambiguity leads to confusion, especially if these descriptions are implemented in more formal contexts such as in engineering knowledge bases. In this paper we sketch a route to overcome this confusion and present the results to which it leads. Our case is the implementation of functional descriptions in Gellish English [4], which is standardized structured English defined in STEPlib [5]. Doing so, we accept the different notions of function and we incorporate details provided by philosophical analyses only in so far this is needed to make these notions precise for engineering purposes. Our results are, first, a proposal of how to implement functional descriptions in knowledge bases such as those expressed in Gellish English. Second, we give practical characterizations of several notions of functions, and an analysis of how these notions are related. Third, and more methodologically, this paper shows how one can strike a workable balance between engineering and philosophy when using a formal language to implement functional descriptions in a knowledge base: philosophy may provide for the conceptual foundation needed to implement ambiguous engineering understandings of functional descriptions; engineering can conversely provide for criteria to prune philosophical accounts to manageable detail.
We introduce, in section 2, four intuitive notions of functions used in engineering, (1) functions as used in systems engineering, (2) functions as used in knowledge base systems, (3) functions as capacities as analyzed in the philosophical literature, and (4) functions as used in design methodologies; the first three of these we will consider in more detail. In section 3, we present the basics of Gellish English, which is a kind of standardized structured English defined by its dictionary or knowledge base called STEPlib. Gellish enables to describe individual objects as well as kinds of objects, such as engineering products and processes in a neutral, system independent and computer interpretable way. It provides a common language for storage of data in databases and for exchange of data between systems. After this inventory, in section 4 we turn to the philosophy of technology. We illustrate the disagreement among philosophers about the analyses of functions as capacities, and, moreover, will show that the engineers' analysis of the notion often lacks the precision and level of detail of that of the philosophers. We consider two proposals taken from the philosophy of technology literature of how to analyze functions. We take stock in section 5, introducing our taxonomy of the main three senses of the function concept in formal Gellish English, and wrap up results in section 6.

2 FUNCTIONAL DESCRIPTIONS

In section 1, we introduced intuitively four notions of function relevant for engineering. The first function notion, we consider, stems from *Systems Engineering* [6]. We will call them *SE-functions*. In systems engineering, functions are discrete actions necessary to achieve the system's objectives [7]. We use verbs, sometimes in combination with a substantive to express those functions. Examples are the decomposition of the function of *disaster control*: such as *blocking the road, remove fire extinguishing water, remove dangerous liquids, keep the flight route unblocked*, etc. We may state these functions explicitly, or we may derive them from the requirements stated. In the end, the functions will be performed or possibly accomplished through a combination of, among other things, personnel, facilities, software, and the use of equipment. What is most important, this use of the function concept leaves out explicitly the underlying material instantiation of the function, and intuitively we may say that the SE-function stresses the activity that is involved when a task is accomplished.

The second notion of function may be found in ontologies underlying formal languages used to build a knowledge base. For instance, in Gellish English each fact is expressed as one or more *relations* between things, where things may be abstract things or even relations. Component_1 is a part of composite_1, is an example of such a fact. An ordinary or static fact is something that is the case without the occurrence of something that changes that situation. Beside the static facts, the Gellish language also covers dynamic facts, or *occurrences*, which are states in which a situation changes. An occurrence, then, is described as an interaction between the things involved in the occurrence. Thus, the expression of an occurrence requires the expression of several facts, each of which describes the role played by some thing in the occurrence. Each thing involved has its own contribution and should be suitable for its role in the occurrence. Gellish covers both kinds of facts and uses two kinds of *elementary facts* to describe the roles that the related objects play in the facts: (1) the elementary fact that something plays a role of a particular kind (in a fact), e.g., an engine plays the role of mover, and (2) the elementary fact that such a role (of that particular kind) is required by a fact, e.g. the role mover is required by a driving process. Returning to the concept of function, we may observe that when we talk about the function of something, we usually do not point to what something is, but to the role it fulfills in a particular occurrence. Moreover, the term function is usually not used to refer to a role in a relation that expresses an ordinary (static) fact. This interpretation of the term function is the reason why in Gellish English the term function became an indication of a *subtype of a role in an occurrence*, as opposed to a role in a relation expressing an ordinary (static) fact. As it turns out, a function is often a subtype of the role of a 'performer', such as, for instance, a driver. We will refer to functions being subtypes of a role in an occurrence as *KB-functions* for knowledge base functions.

The third way to analyze the notion of function we take from the philosophical literature on the subject of functions. There, functions are primarily connected to the role of an *ability* or *capacity*: if an object or process has the ability or capacity to execute some task, then this ability has the role of a function within the possession relation with its possessor. A well-known proposal to explain functions as capacities, is that of Robert Cummins [8]. According to this proposal, which we will discuss in more detail in section 4, functional properties are always ascribed to objects or processes relative to some capacity of some containing system. More precisely, functional ascriptions are relative, to an analytical account of a decomposition of the containing system's capacity into sub-processes, which explain the capacity. For instance, according to Cummins we may claim that the heart functions as a pump, provided we put it in context of an account of the capacity of the circulatory system to transport all kinds of substances such as food, oxygen, wastes, etc. Moreover, this account must hinge on the pumping capability of the heart. Since this background is the usual one, it makes the statement "The heart functions as a pump" sounds appropriate, and makes the statement "The heart functions as a noisemaker" sound false, even without the background account just mentioned. Thus, in contrast to the use of the function concepts in systems engineering and knowledge bases, function often refers in philosophy to the role of an *ability* or *capacity*: if an object or process has the ability or capacity for some task, then this ability has the role of a function within the possession relation with its possessor. We will refer to this notion of function with *C-function.*

Finally, to encounter a fourth use of the function concept we could also have turned to design methodology. There, functions are sometimes taken to be *purposes* (e.g., [9], [10]), which implies that much *means-end reasoning* [11] is involved. We will not consider this meaning of the term separately in the present paper since purposes are already separately accommodated in Gellish English.

3 GELLISH ENGLISH

Gellish English is a structured subset of natural English that can be stored in standardized database tables. It is an extension and combination of the ISO standards 10303-221 (AP221) and ISO 15926. Knowledge paraphrased in Gellish English forms a knowledge model that differs from conventional data models in information science, because of the flexibility and general applicability of the Gellish data structure. It is flexible because its application scope and semantic expression capabilities can be extended without a modification of the data model and it is generally applicable, because it can be integrated with embedded domain specific knowledge from any discipline area. Although its terms are from natural language, Gellish may also be called an artificial language that does not define its own vocabulary. The Gellish English dictionary extends a standard English dictionary in that it contains additional knowledge that is expressed as computer interpretable relationships between the concepts. This makes the Gellish dictionary a 'smart dictionary' or knowledge base.

Let us turn to some core concepts used in Gellish English. Any fact in the real world, or anything that is (thought to be) the case is expressed in Gellish as a relation or relationship that relates two or more things, each of which has a particular role in the relation. Note, by the way, that we use the terms relation and relationship as synonyms. Consequently, any single fact is expressed in Gellish as a binary, ternary or higher order relationship. Thus, the core concept of the Gellish data structure is a relationship. For example, the fact: 'the Eiffel tower is located in Paris' is a relation between the Eiffel tower and Paris, whereas the relation type is of the kind 'is located in'.

The Gellish data structure is intended to enable the expression of any kind of fact, at least in the engineering world. Basically, it consists of a generic structure for relations that is suitable to express any kind of 'fact'. That basic structure is presented in the next figure.

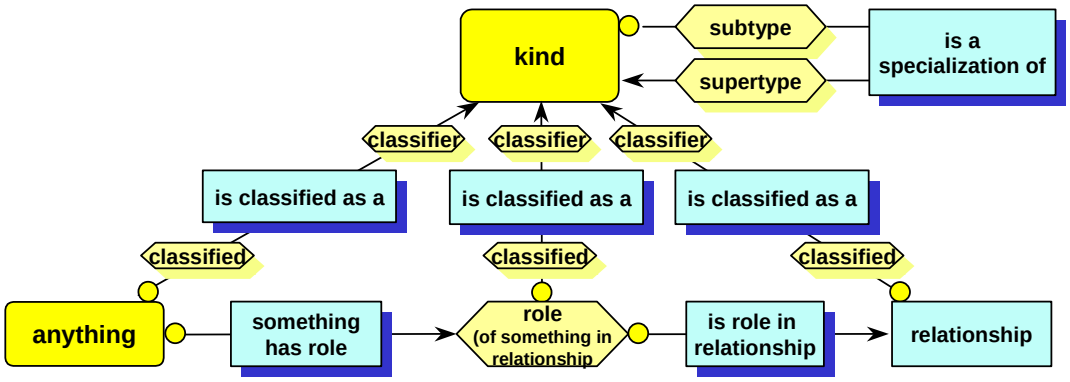

Figure 1. The generic structure to express a fact

To interpret the meaning of an expression of a fact, it suffices to consider some instances of the structure of Figure 1. For example, the Eiffel tower and Paris are two instances of 'anything' each of which has its own role in a relationship of the kind 'is located in'. When we convert the graphical representation of the expression to a number of lines in a Gellish Table, then the meaning of the expression becomes computer interpretable, especially when that includes lines that classify 'Eiffel tower' as a 'tower' and 'Paris' as a 'city', while tower and city are standard concepts in the Gellish smart dictionary.

The various kinds of relationships are standardized in Gellish and they are the core elements that determine the expression power of the Gellish language. The relation types form a specialization hierarchy (subtype-supertype hierarchy) of relation types. The relation types are further defined by the kinds of roles that they require. For example some binary kinds of relationships with rather trivial kinds of roles are:

- A composition relationship (A is a part of B), with the roles 'part' and 'whole',
- A classification relationship (A is classified as a C), with the roles 'classified' and 'classifier'.
- A specialization relationship (C is a specialization of D), with the roles 'subtype' and 'supertype'.

The kinds of roles determine the kinds of things that are suitable to play those kinds of roles, because only specific kinds of things can play specific kinds of roles. For example:

- Each ‘part’ role and each ‘whole’ role in a composition relationship can be played only by an individual thing,
- Each ‘classified’ role in a classification relationship can be played only by an individual thing and each ‘classifier’ role can only be played by a class (being a kind of thing).

Gellish makes a distinction between *ordinary* relationships and *elementary* relationships. An ordinary relationship is a complete relationship used to express a complete fact. Examples of ordinary relationships are the above mentioned composition relationship and the classification relationship. Elementary relationships are used to express the internal structure within an ordinary relationship. There are two kinds of *elementary* relationships to express actual facts:

- A ‘has as role’ relationship,
- A ‘is played by’ relationship,

and their two conceptual elementary counterpart relationships to express knowledge:

- A ‘requires as role a’ relationship,
- A ‘can be played by a’ relationship.

Any ordinary kind of relationship can be *defined* by repeated usage of the two conceptual elementary kinds of relationships. This is illustrated in Figure 2.

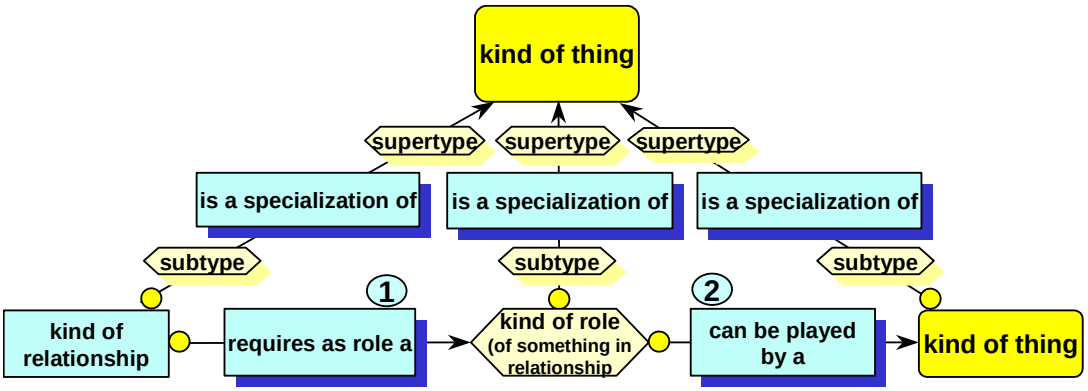

Figure 2. Definition of a part of a kind of relationship

Figure 2 presents a structure used to express a definition of a part of an ordinary relationship. (Note the similarity with Figure 1.) Such a structure is required to define the parts of any ordinary kind of relationship. The two elementary kinds of relationships (indicated in Figure 2 by the numbers 1 and 2) express two elementary kinds of facts. The first two rows of Table 1 show how those facts are expressed in the Gellish language. Any ordinary *binary* relationship requires two roles, a first role and a second role, each of which is played by a thing. Thus a full expression of a binary fact requires two partial expressions each of which conforms to Figure 2.

Table 1

Fact id	Left hand term	Relation type	Right hand term
1	kind of relationship	requires as role a	kind of role
2	kind of role	can be played by a	kind of thing
3	*A*	is a part of	*B*
4	*B*	is a whole for	*A*

It is important to note that *the natural language phrase representing the kind of relationship determines which kind of role acts as the first role and which kind of role acts as the second role.* For example, assume there is a fact that can be expressed in Gellish by the assembly relation on row 3 in Table 1, then that expression is equivalent to and expresses the same fact as the expression on row 4 in Table 1. According to the above normal English convention both expressions imply that object *A* has a role as part and object *B* has a role as whole in a relationship that is classified as an assembly relation which is also called a ‘part-whole’ relationship.

Table 2

Fact id	Left hand term	Relation type	Right hand term
5	relation	is a specialization of	anything
6	relation	requires as first role a	relator
7	relator	can be played by a	anything
8	relator	is a specialization of	role
9	relation	requires as second role a	related
10	related	can be played by a	anything
11	related	is a specialization of	role
12	assembly relation	is a specialization of	relation
13	assembly relation	requires as first role a	part
14	part	can be played by a	individual thing
15	part	is a specialization of	relator
16	assembly relation	requires as second role a	whole
17	whole	can be played by a	individual thing
18	whole	is a specialization of	related
19	individual thing	is a specialization of	anything
20	is a part of	is a synonym of	assembly relation
21	is a whole for	is an inverse of	assembly relation

The most generic kind of relationship in Gellish is simply called 'relation' or 'might be related to'. That concept forms the top of the subtype-supertype hierarchy (taxonomy) of relation types. The definition of that 'relation' concept is defined in Gellish by the seven expressions as presented on the first seven rows of Table 2 (facts 5-11). The order of a relation type is defined in Gellish by the cardinalities of the roles in the relation. An example of a lower level relation type is given on the lower rows of Table 2 (fact 12-19), which show the definition of a (binary) assembly relation. The assembly relation is defined to be a subtype of relation and is distinguished from its supertype and from other neighbor subtypes by the fact that it has particular types of roles which together with their role players are defined by two pairs of elementary facts, each of which pair is expressed conform Figure 2. The structure of Figure 2 is a basic building block to define or express any kind of fact, in other words: to express knowledge. Two of such building blocks are required to express a binary fact, three building blocks are the basis for a ternary fact, etc.

To end this section we will look at some other essential Gellish phrases. Gellish enables the definition of synonyms for names of concepts as well as for names of kinds of relationships. For kinds of relationships there are Gellish 'phrases' defined as standard synonyms. Synonyms are often the preferred terms or phrases in a particular context or language. For example, standard synonyms are defined in the Gellish dictionary as in fact 20 in Table 2. Gellish has also defined inverse Gellish phrases that imply that the left hand and the right hand related objects are inversed. Fact 21 of Table 2 shows an example.

The following relation types are used to express facts that occur in the description of functions: the *possession of aspect* relation and the *involvement in occurrence* relation, with its subtypes *performer of occurrence* relation and *subject of occurrence* relation. Table 3 displays examples of usage of the Gellish phrases for those relations in a fact that state that an engine has an ability to drive (a pump) and is actually driving a pump.

Table 3

1	engine	is a possessor of	ability to drive	with roles *possessor* and *possessed*
2	engine	is a performer of	driving a pump	with roles *performer* and *performed*
3	pump	is a subject in	driving a pump	with roles *subject* and *subjecting*

4 PRUNING PHILOSOPHICAL C-FUNCTIONS

In philosophy analyses of the concept of function were originally focused on the biological functions and have only recently lead to also a systematic analysis of the technical counterpart [12], [13]. Of the

various accounts of technical functions that this analysis has produced, we present two that stay close to engineering practices and on which functions are identified with capacities of artifacts. The first is the account of Cummins [8], which we already discussed in section 2. The second is the ICE-function theory [14], [15]. A first look at the definitions of these two accounts leads, however, quickly to the conclusion that the precision provided by philosophy may hinder formalization, rather than facilitates it. The definition of functions as given by Cummins is as follows ([8], p. 762):

> *x functions as a ϕ in s (or: the function of x in s is to ϕ) relative to an analytical account A of s's capacity to ψ just in case x is capable of ϕ-ing in s and A appropriately and adequately accounts for s's capacity to ψ by, in part, appealing to the capacity of x to ϕ in s.*

The relevant definition in the ICE-theory is about ascribing technical functions, and reads ([15], p. 69):

> *An engineer e ascribes the capacity to ϕ as a function to the component c, relative to the plan to compose c, c′, c″, … in configuration k in order to obtain an artefact x with the capacity to ψ, and relative to an account A, iff:*
> *I. the engineer e has the capacity belief that c has the capacity to ϕ in the configuration k, and the engineer e has the contribution belief that if x has the capacity to ψ, this is due, in part, to c's capacity to ϕ;*
> *C. the engineer e can justify these two beliefs on the basis of A; and*
> *E. the agents d who developed the plan to compose c, c′, … in configuration k, in order to obtain an artefact x with the capacity to ψ, have intentionally selected c for the capacity to ϕ and communicated this plan to other agents u.*

The first definition is relatively simple and introduces the additional notions of analytical account and explanation to the analysis of functions: functions, colloquially put, are by this definition those roles ("to ϕ") of items *x* that explain on the basis of some account *A* why larger systems (*s*) have specific capacities ("to ψ"). Compared to the intuitive notion of functions as roles, this definition thus introduces a constraint: a role that can count as a function should be relevant to a capacity of a thing or process. But this constraint is made relevant to a body of knowledge: an account is needed to single out those roles that are relevant. The introduction of a constraint may be welcomed for our formalization, but its relation to an account seems less attractive. In various philosophical analyses of functions the inclusion of these accounts are sometimes motivated by a wish to include into the analysis functions ascribed by types of knowledge different to science and technology (say, homeopathic knowledge or pre-Newtonian knowledge). From an engineering perspective this liberalism seems, however, redundant, suggesting that the account *A* can be taken as by default consisting of scientific and technological knowledge only.
The second definition of the ICE-theory introduces in turn an access of additional notions, and even is, strictly speaking, not giving criteria under which it can be said that items have a technical function, but giving criteria when we are *allowed to ascribe* functions to items. Compared to the first definition it introduces an even stronger constraint: a role that can count as a function of a component should contribute to a capacity of the artifact of which it is a part *and* be designed for this contribution. And although also this constraint may be helpful for accommodating functions in formal languages, it is again accompanied by a series of new concepts that seem less helpful. In an attempt to single out only the helpful elements, we again tried to limit these additional concepts. We, for instance, suppressed the references to engineers that ascribe the functions by assuming that the ICE-theory can be interpreted as being about components having functions and by assuming that conditions I and C are automatically fulfilled by successful designing. The "pruned" definitions we obtained in this way has for Cummins' account the form:

> *the function of x in s is to ϕ just in case x is capable of ϕ-ing in s and the capacity of x to ϕ in s contributes to s's capacity to ψ.*

and for the ICE-theory the form

> *the component c has the capacity to ϕ as a function in artefact x with the capacity to ψ, iff: the agents d who designed x included c because they believed that c has the capacity to ϕ in x, and they believe that the capacity to ϕ of c contributes to x's capacity to ψ.*

In Figure 3 and Figure 4 we give the corresponding diagrams.

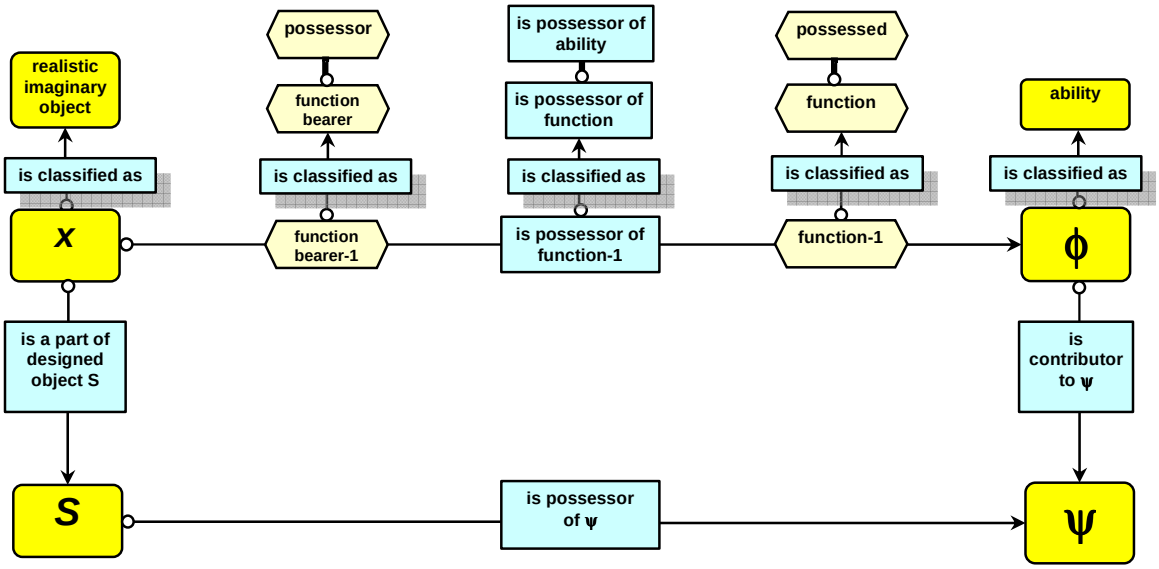

Figure 3. Cummins' functions

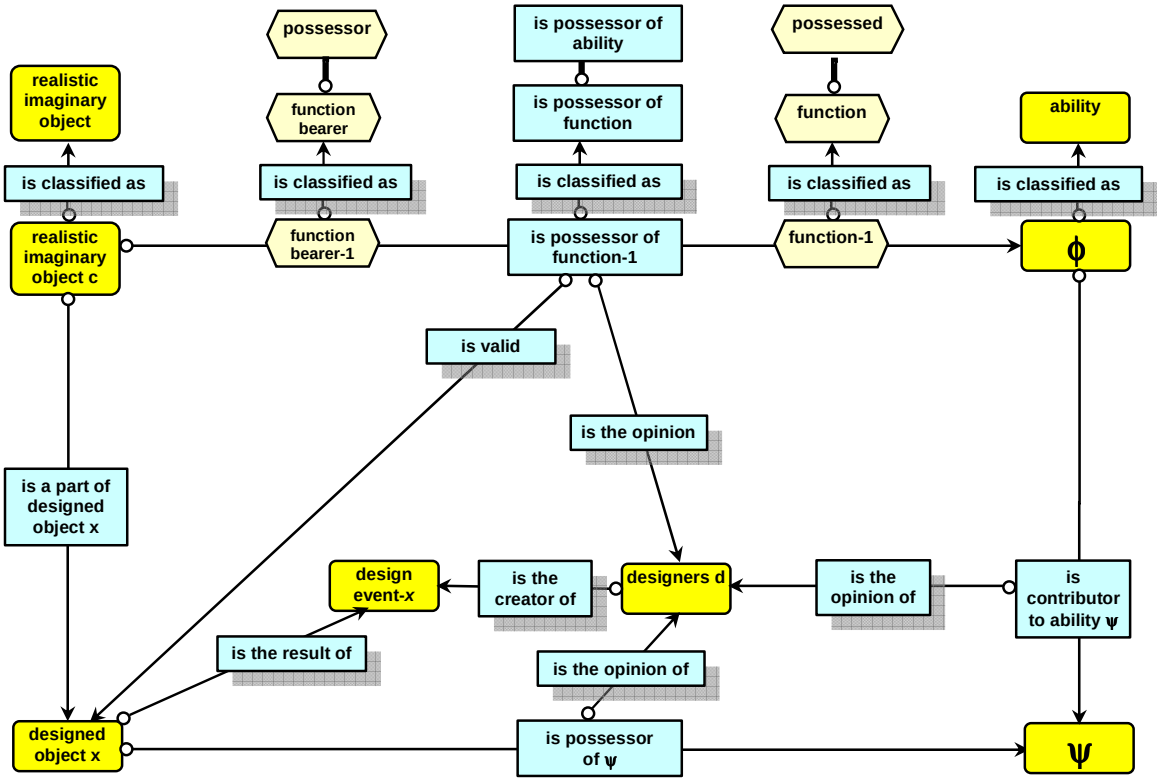

Figure 4. ICE functions

5 THE GELLISH TAXONOMY OF THREE FUNCTION CONCEPTS

With the intuitive explanations of various function concepts, and some knowledge of our formal tool at hand, we will sketch in this section the outcome of our search for a formal definition of the function concept used in engineering. When looking for such a definition, we soon encountered two problems. The first problem was the absence of a general consensus within the engineering and design methodology community about the content of their functional descriptions as illustrated by the different intuitive meanings we discussed in section 2. The second problem was the large gap between

the level of precision between the engineering characterizations of functional descriptions and that of philosophers. To illustrate this point, the ISO 15288 norm, which sets the current standard for systems engineering, does not even define the notion of a function and therefore lacks the precision required by formalization. As for the first problem, the lack of consensus, our view is that it may be taken as deplorable but it does not need to block formalization; if the ideal of a uniform and general meaning has to be abandoned, one still can formalize the different notions of functional descriptions as long as they are carefully distinguished. In this section, we will develop a taxonomy of how the three function concepts mentioned in the first section are related, using the rigorous formalism of Gellish English as an overarching framework.

As to the difference in detail of the engineering proposals and those of philosophers, we looked for a pragmatic balance. This difference is understandable, of course, given the different aims of engineers and philosophers. For engineers a general indication of how to use terms may guarantee colloquial communication; for philosophers elaborated definitions are needed given their habit to challenge any conceptual clarification by all kinds of problematic counterexamples. Yet, when formalizing functional descriptions for enabling more formal communication in engineering, some but not all details made available in philosophy are useful. Hence, what is needed is that philosophical analyses of functional descriptions are simplified down to a level of detail reasonable from an engineering perspective, and we discussed how to do that in the previous section.

As mentioned before, Gellish English distinguishes between static facts and occurrences, i.e., dynamic facts, whereas facts are expressed as relations between two or more things. These things may be material things but also immaterial things like properties or capacities. Thus the first flavor of functions, the C-functions, may be appropriately expressed as a fact, viz. one individual physical object has the ability or the capacity to ϕ. For example, a pump may be driven by an engine taken as a physical object. Of course the engine need not always perform this ability, but in the right circumstances it may be able to perform this task. We may say that the physical object, the engine, has the capacity or C-function to drive the pump, irrespective of that is the question whether the engine and the pump are connected in a suitable way. In Gellish, the fact that a physical object has some ability or capacity is expressed as a relation between this object and the capacity; and the relation is the possession relation. This state of affairs is depicted as an arrow from the physical object to the ability in case on the right hand side of Figure 5. The rectangle in the middle of this arrow qualifies the relation to be one of possession. The hexagonal box on the arrow at the side of the small circle represents a first role (role-1) that is played by the left hand object. Thus the role of function bearer is played by the physical object in the possessor relation. The hexagonal box at the side of the arrow point represents the role played by the object on the right hand side. In our case, it indicates that the ability or capacity has the role of the C-function in the possessor relation between the physical object and the ability. The boxes connected with the fat lines and the hollow bullets above the five text boxes mentioned already indicate the generic terms (the supertypes) covering the concepts just explained. Thus, the ability is a kind of (is a subtype of) quality and the physical object is a kind of individual thing. Furthermore, it should be noted that the C-function as located in Figure 5, only is an outline of the capacity or ability function as defined by Cummins. Here, we take a C-function to be a capacity or ability used by the object to achieve a specific end. In the previous section we saw how the complex version of the C-function can be pruned into a simplified version of the C-function that suits our purpose to locate it in our taxonomy of functions formulated in Gellish English.

To get grips on the KB- and SE-functions let us consider the same example of a pump that is driven by a physical object, the engine. The engine driving the pump is a dynamic fact so the KB-function has to be expressed in the Gellish occurrence “engine driving the pump”. As the SE-function is an activity it is also to be expressed in an occurrence. The question arises, then, how to connect both functions mentioned to the occurrence. Intuitively we saw that a KB-function is expressed as being a (subtype of) role in an occurrence. Often the subtype is the performer role. The KB-function in our driving pump example is to have the performer role of the driving. More generally, we could say that the KB-function is to have the *x* role of the action in the occurrence. This explains the location of the KB-function in Figure 5. The KB-function is the performer subtype of the role of the physical object engaged in the occurrence of driving a pump—and so indeed the function is being the performer in the action of driving. This function concept is apprehensible from the knowledge base perspective. When a knowledge base is build, the specific object carrying the roles already exist, or will come into existence, and the task of the knowledge base builder is to put all relevant descriptions of relations into

place. That is the reason for the KB-functions being closely related to objects or things performing tasks.

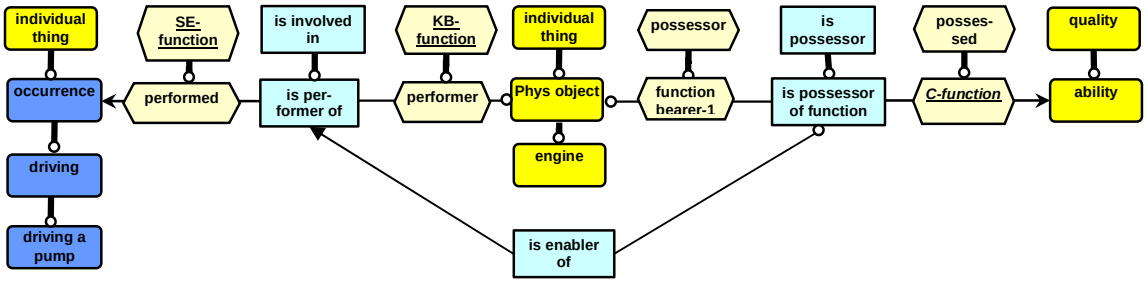

Figure 5

We saw intuitively that SE-functions are activities that have to be carried out. Also from the perspective of designers, the focus is more on the activities that have to be carried out than on the objects or systems that perform these activities. Thus, blocking the road, counts as an example of a SE-function, or in our example of the pump, intuitively, the action of driving the pump, is the SE-function. More precisely, when an object is a performer in an occurrence, then, the SE-function is the "performed" subtype of the action's role. This is clearly indicated in Figure 5. The physical object is in "performer of" relation with the occurrence, and the SE-function is the role of the occurrence, i.e., the action that is "performed". Untying the knot of the term function in engineering contexts, Figure 5. shows how the three meanings of the term are mutually related.

6 CONCLUSIONS AND RESULTS

The exercise to use Gellish English to disentangle the Gordian knot of the various meanings of the function concepts has led to three main results. First, using the knowledge base language Gellish English we singled out at least three different meanings of the term function in engineering contexts. A *C-function* is the possessed ability or capacity of the performing objects, a *KB-function* is being the performer of an activity, and an *SE-function* is the performed action. In the engine-driving-the-pump example the C-function is the capacity of the engine to drive a pump, the KB-function is being the driver of the pump, and the SE-function is the performed driving of the pump.

Distinguishing between these three meanings will be directly relevant to the ISO/IEC 15288 standard on Systems Engineering. This standard states as its purpose that it "establishes a common framework for describing the life cycle of systems created by humans" and makes extensive use of the terms 'function', 'functional' and 'functionality' (38 times). Nevertheless it does not contain a definition of what a function is. The current analysis makes explicit what these functional terms may mean in that ISO standard, and resulted in a proposal for such a definition in the next version of that standard. This may make the implementation of that standard in computers less ambiguous.

Distinguishing between C-functions, SE-functions, and KB-functions is also beneficial to projects in which engineers model their knowledge. For example, the Dutch organization for Infrastructure, Public space and Transport, CROW, uses the results of our research when modeling their knowledge and putting it in an object library. Distinction between these concepts and their incorporation in the training material resulted in less discussions, better definitions, less ambiguity and better computer interpretable descriptions. Many functions were originally defined as abilities of systems and components (C-functions). For example, it was originally specified that a sewer system has transport of rainwater as its function. This transport of rainwater was *defined* as an ability of a sewer system. However, after getting clarity about the various usage of the term function, it appeared that transport of rainwater was not meant as an ability of a sewer system, but as an occurrence that needs to take place, irrespective of the question whether a sewer system should be applied or not.

Second, we showed how the complex definitions of function found in the philosophy of technology literature, can be applied in engineering if pruned down to the level of complexity appropriate for implementation in formal languages.

To introduce the third result, we must realize that a functionality of knowledge bases that would be very helpful for designers and system builders would be the ability of making queries to the effect of

finding out which combination of objects would fulfill some on beforehand specified function. To achieve the implementation of such queries, all three concepts will be included in the Gellish language, taking SE-functions as the preferred meaning of function.
Our fourth result is that we not only singled out three different meanings of the term function, but also related these meanings at least for functions of physical objects in formal Gellish English (see section 5 and in particular Figure 5). In principle this would allow us to translate each of these three meanings, say, a SE-function, into the other two, i.e., into a KB-function and a C-function: if an engineer describes in Gellish English the driving of a pump by giving all the relevant relations as depicted in Figure 5, then software that can interpret Gellish English can easily generate the related KB- and C-function of that pump. Whether this fourth result holds in general, e.g., when functions of processes or activities are considered rather than those of physical objects, is a topic for future research.

REFERENCES

[1] Chittaro L. and Kumar A.N. Reasoning about function and its application to engineering. *Artificial Intelligence in Engineering*, 1998, 12, 331-336.

[2] Chandrasekaran B. and Josephson J.R. Function in device representation. *Engineering with Computers*, 2000, 16, 162-177.

[3] Kitamura Y. and Mizoguchi R. Organizing knowledge about functional decomposition. In Folkeson A., Gralén K., Norell M. and Sellgren U. (eds.), *Research for Practice: Innovation in Products, Processes and Organisations. Proceedings of the 14th International Conference on Engineering Design, August 19-21, 2003*, abstract: pp. 55-56, full paper on accompanying CD-ROM, 2003 (Design Society).

[4] Van Renssen A.S.H.P. Gellish: a generic extensible ontological language: design and application of a universal data structure, 2005, Delft University Press, Delft. https://www.darenet.nl/en/page/language.view/search.page.

[5] STEPlib part 0-4: http://sourceforge.net/project/showfiles.php?group_id=28353&package_id=20196

[6] ISO/IEC 15288:2002 Systems Engineering – System Lifecycle Processes. http://www.iso.org/

[7] *Systems Engineering Fundamentals* January 2001 Defense Acquisition University Press, Fort Belvoir. p.45

[8] Cummins R. Functional Analysis. *Journal of Philosophy*, 1975, 72, 741-765.

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

[10] Gero J.S, Tham K.W. and Lee H.S. Behaviour: a link between function and structure in design. In Brown D.C., Waldron M.B. and Yoshikawa H. (eds.) *Intelligent Computer Aided Design*, pp. 193-225, 1992 (Elsevier, Amsterdam).

[11] Hughes, J., Kroes P. and Zwart S.D. A Semantics for Means-end Relations. *Synthese*, DOI 10.1007/s11229-006-9036-x.

[12] Kroes P.A. and Meijers A.W.M. (eds.) *The Empirical Turn in the Philosophy of Technology*, 2000 (JAI: Amsterdam).

[13] Kroes P.A. and Meijers A.W.M. The dual nature of technical artifacts. *Studies in History and Philosophy of Science*, 2006, 37, 1-4, and the other contribution to this special issue.

[14] Vermaas P.E. and Houkes W. Technical functions: a drawbridge between the intentional and structural nature of technical artefacts. *Studies in History and Philosophy of Science*, 2006, 37, 5-18.

[15] Vermaas P.E. The physical connection: engineering function ascriptions to technical artefacts and their components. *Studies in History and Philosophy of Science*, 2006, 37, 62-75.

Contact: S. D. Zwart
Delft University of Technology
Department of Philosophy
Jaffalaan 5
NL-2628 BX, Delft
The Netherlands
Phone: + 31 (0)15 2785906
Fax: + 31 (0)15 2786439
e-mail: s.d.zwart@tudelft.nl