Proceedings of the ASME 2007 International Design Engineering Technical Conferences&
Computers and Information in Engineering Conference
IDETC/CIE 2007
September 4-7, 2007, Las Vegas, Nevada, USA

DETC2007-35373

TOWARDS A REFERENCE ONTOLOGY FOR FUNCTIONAL KNOWLEDGE INTEROPERABILITY

Yoshinobu Kitamura **Sunao Takafuji** **Riichiro Mizoguchi**

The Institute of Scientific and Industrial Research
Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{kita, takaj, miz}@ei.sanken.osaka-u.ac.jp

ABSTRACT

Functionality is one of the key aspects of artifact models for design. A function of a device, however, can be captured in different ways in different domains or by different model-authors. Much research on functions has been conducted in the areas of engineering design, functional representation and philosophy, although there are several definitions and notions of functions. We view conceptualization of function is multiplicative in nature: different functions can be captured simultaneously from an objective behavior of an artifact under different teleological contexts of users/designers, or from different viewpoints (perspectives) of a model-author. Such differences become problematic for sharing functional knowledge among engineers.

In this article, we attempt to clarify the differences of such perspectives for capturing functions on the basis of the ontological engineering. On the basis of a generalized model of the standard input-output model in the well-known systematic design methodology, we show descriptive categorization of some upper-level types (classes) of functions with references to some definitions of functions in the literature.

Such upper-level ontological categories of functions are intended to be used as a reference ontology for functional knowledge interoperability. One of the two usages here is to convert functional models between different functional taxonomies. A functional term in a taxonomy is (ideally) categorized into a generic type defined in the reference ontology. It is widely recognized in the literature that such an upper-level ontology helps automatic 'mapping discovery' which is to find similarities between two ontologies and determine which concepts represent similar notion. The reference ontology of function might have such an effect. Another usage of the reference ontology is to integrate fault knowledge into functional knowledge and automatic transformation of FMEA sheets. The designer can describe an integrated model of both functional knowledge and fault knowledge. Based on ontology mappings, automatic transformations of FMEA sheets can be realized.

In this article, we discuss the detail of the definitions of the upper-level categories of functions ontologically. Then, we give an overview of usages and effects of the upper-level categories as a reference ontology for functional knowledge interoperability.

KEYWORDS

Functional representation, ontology, design knowledge management, functional design

1. INTRODUCTION

Functionality is one of the key aspects for capturing technical artifacts. Although much research has been conducted on the representation of functionality in functional representation [4][10], engineering design [3][14][27][33][35][38][42], philosophy [16][34] and value engineering [29], there is no common definition of functions [4][9][38]. The remaining ambiguity about conceptualization behind the definitions makes knowledge-authors to describe functional knowledge in an ad hoc way. For example, a function of an electric fan can be represented as "to move air", "to cool human body", or "to make human comfortable". Such different conceptualizations of functions are caused by, we believe, difference of viewpoints (perspectives) for capturing functions. In the example, the conceptualization of "to move air" focuses on the input and output airflows of the fan, while "to cool human body" focuses on the interaction between the fan and the user. Depending such a *capturing perspective*, a function is captured in different ways. In our experience of successful industrial deployment of functional knowledge management in manufacturing companies [19], there are many cases that the capturing perspective is not consistent and confused for a functional model.

Our aim here is to clarify the relationship among such several different conceptualizations of functions in terms of the *capturing perspectives*. In this article, we identify some of elements of viewpoints for capturing functions such as the spatial location of focus of attention and the time interval of effects of function. Using these capturing perspectives, we identify some upper-level types (categories) of functions such as "flowing-object function" and "environmental function". We categorize

them hierarchically as an ontology.

The fundamental usage of such upper-level concepts is to help the model-authors capture function consistently. Many of the concepts in the ontology are based on real and confused functional models written by engineers in the deployment. They justify needs of such concepts.

Such categorization is also intended to be used as a reference ontology for knowledge interoperability. Ideally, a concrete function is categorized into a type in the reference ontology. For example, the "to cool human body" function of the electric fan is categorized into the "environmental function" type in the reference ontology. Note that it is not our goal to build such an ontology that subsumes all the functions of engineering products, i.e., a super-set or a standard set of functions. Rather, we intend to classify a variety of functions into large sets with ontological differences. The reference ontology can be used for *mapping* among functions, for example, in different functional taxonomies. Such ontological mapping facilitates interoperability of knowledge.

In Section 2, as a starting point, we discuss our device-oriented definition of function [19]. This is based on the input-output relation model that is similar to the well-known model in the German-style systematic design [33]. We analyze characteristics of function and define function in relationship with objective behavior under a specific teleological context.

Next, we generalize the input-output model into a generic effect model in which an operand is affected and changed by an agent. We identify types of the elements in the generalized model such as operands, their location, time intervals of the change, etc. Using the types of the elements, we categorize types of functions.

Then, we discuss how to use the generic categories of functions. After fundamental usage as prescriptive guidelines for capturing functions, we discuss two usages of the generic categories of functions as a reference ontology for interoperability of different knowledge forms. It can be used for integration of fault knowledge with functional knowledge and interoperability between different functional taxonomies.

This article is a generalized (upper-level types) and/or detailed version of the ontologies in our other papers [18][19]. As applications, the interoperability between functional taxonomies is under investigation and partly is presented elsewhere [32]. Section 6.2 generalizes our previous framework [20].

2. BEHAVIOR, FUNCTION, AND CONTEXT

For clear definition of functionality, the relationship with "behavior" plays a crucial role. The distinction between function and behavior is discussed in areas such as qualitative reasoning (e.g., [7]), engineering design and philosophy. The relation between behavior and function is so-called as "means and ends" [23], F-B relationship [42] or "aims-means" [14].

A behavior of an engineering system can be viewed as an aggregation of components' behaviors. A component's behavior can be represented as "input-output" relation. This view originates from the systems theory, and has been widely adopted in engineering task including design such as the German-style design methodology [33]. A system, a sub-system or a component can be conceptualized as a "device" that behaves. As shown in Fig. 1, a device operates on other things (we call operands) and thus changes their physical states. The operand is something flows through the device via ports and is affected by the device. Examples of the operands include fluid, energy, motion, force, and information. Thus, behavior here represents temporal changes of physical quantities of the operand. The change is represented as a pair of physical-states as input and output of the device, each of which represents a value of a physical-attribute at a port of the device (see Fig. 1). Thus, the behavior is objective and constant in any situation/context.

In comparison with behavior, function is related to a context of use (i.e., teleological) and hence context-dependent. We view a behavior can perform different functions according to contexts of use. For example, a heat exchanger can be used as a heater or a radiator as shown in Fig. 2. 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. This difference of functions depends on the embedded system. The former is, for example, embedded in a power plant and of which the heat-destination-side is connected with a turbine. The latter is embedded in a car with an engine. Moreover, a function can be performed (realized) by different behaviors. A function is associated with specific constraints or properties (*capacity to perform a function*) on a part of behaviors to realize the function. A behavior can perform multiple functions simultaneously. Thus, a function here is rather separated from a device. In [7], No-Function-In-Structure principle is proposed as a fundamental modeling principle, that is, a local model of a device should not include function.

On the basis of the discussion above, we define a (base-) function as follows [17];

A (base-)function performed by a device is a role played by the behavior of the device to achieve a specific goal under a context of use, based on a certain capability inherent in the device.

By role we here mean such a concept that an entity plays in a specific context and cannot be defined without mentioning external concepts [39]. A role is anti-rigid (i.e., contingent

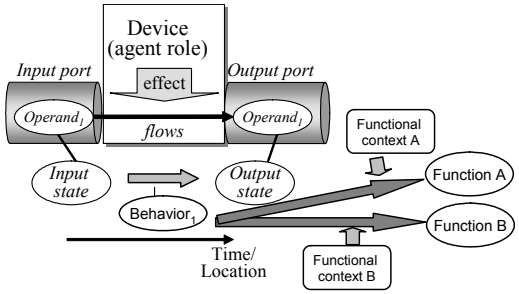


Figure 1. A basic model of a device, operand, behavior and function

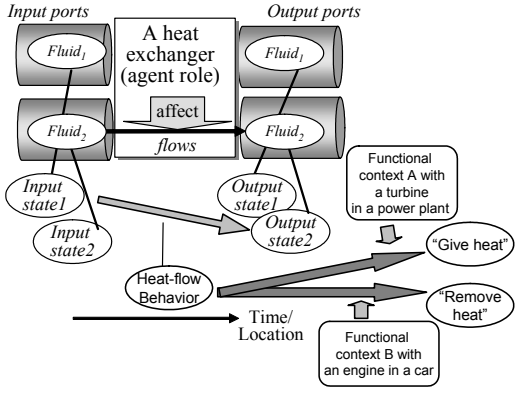


Figure 2. An example of a heat-exchanger's behavior and functions

(non-essential) property for identity), dynamic (temporary and multiple), and founded (i.e., extrinsic property defined with external concept) [28]. According to the observation above, function can be defined as a role. Firstly, a function (and a behavior as its basis) is founded, since a function of a device affects an entity other than the device itself (operand) and causes its temporal changes. Secondly, a function is anti-rigid for a behavior, since a behavior can perform different functions according to contexts without loss of its identity. Thirdly, a function can be performed (realized) by different behaviors. A behavior can perform multiple functions simultaneously. Thus, a function is dynamic and multiple.

We say that "a behavior plays a function role". If a device performs a behavior and the behavior plays a function role in a context, then the device plays a function-performer role in the context. For example, the heat-exchange behavior plays the removing-heat function role and then a heat exchanger plays the function-performer role of removing-heat as a radiator.

The context of use represents teleological goals to be achieved by the function (we call functional context). If the device is the whole product, its functional context (and its goal) is determined by how it is used by users (we call *external (or user) function context*). A function in an external/user function context (we call *external function*) is intended by a user. On the other hand, if a function is of a component embedded in a system, the function contributes to achievement of the function of the system (we call this *component function*). Thus, its function-context (we call *system function context*) is determined by a functional structure, in which the system's function is achieved by a sequence of components' functions. The *system function context* is related to relations in the literature such as function decomposition [33], whole-part relation [23] and degree of complexity [14].

In the conceptual design phase, real devices are usually not determined yet and the functional context can be vague. In our functional model, a (base-)function can exist alone without a device which performs the function and without a full description of a functional context. Under a vague functional context, a function has just a possibility of realization. At the end of the conceptual design phase, a functional structure (functional contexts) is usually determined. A detail of the devices usually will be determined in the detailed design phase.

3. UPPER-LEVEL CATEGORIES OF FUNCTIONS

3.1. A Generic Model of Roles of Effect in Contexts

The definition of function in Section 2 is based on the input-output model from the device-centered viewpoint. By generalizing the input-output model, we can identify several types of functions. Figure 3 shows an effect model generalized from the basic model shown in Fig. 1. The generalized model consists of the basic elements such as *effect* (as a generalized type of *behavior*) and *goal-oriented context* (as that of *functional context*) for representation of generalized function. We categorize subtypes of these elements, which we call *descriptors of functions*

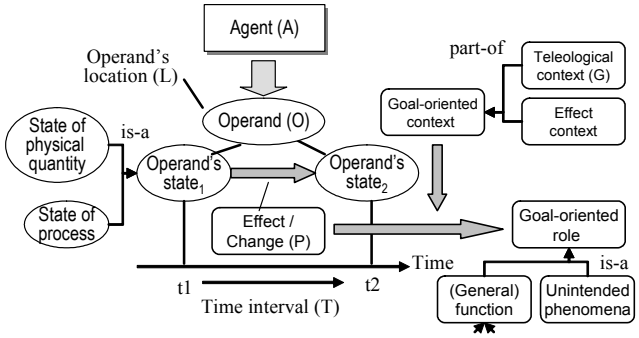


Figure 3. Generalized basic model of agent, operand, effect, context and goal-oriented role

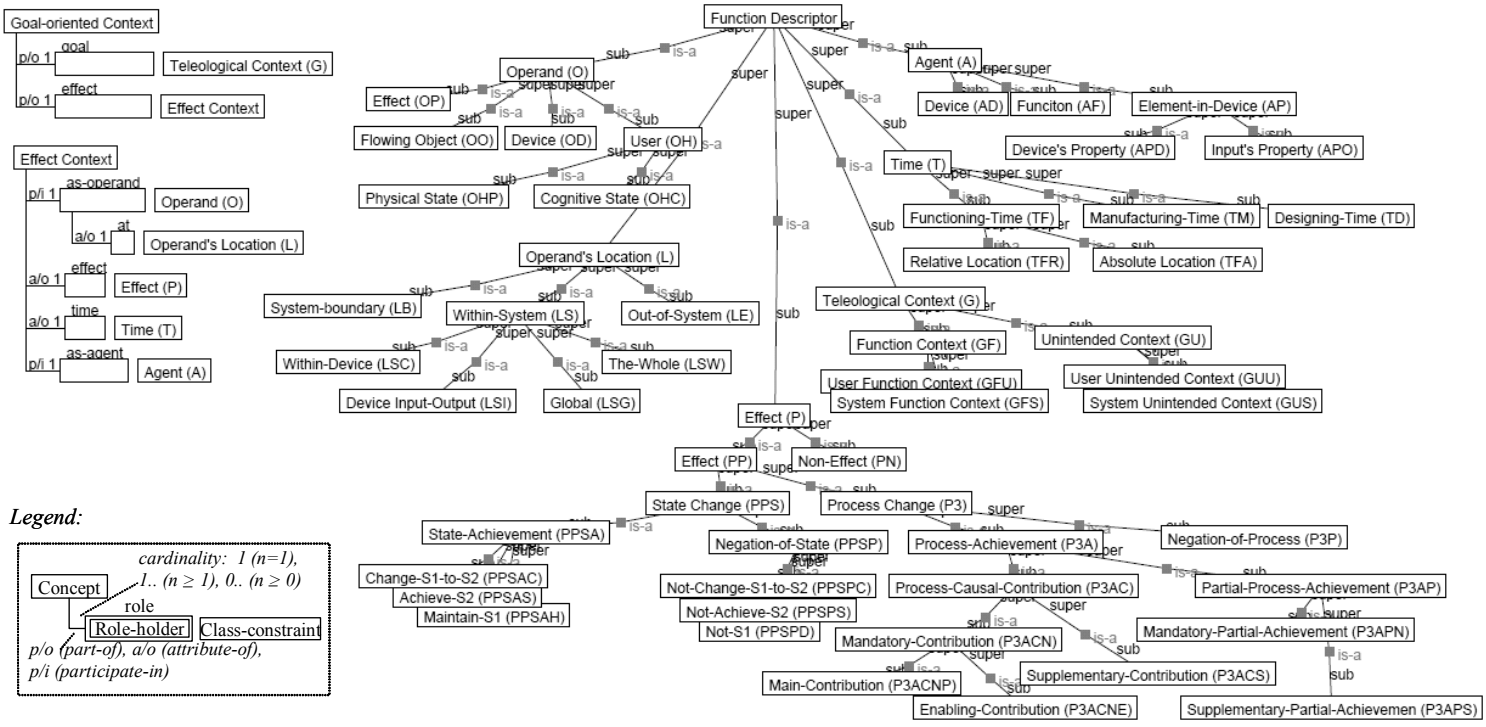


Figure 4. Function descriptors for the categorization in Fig. 5.

shown in Fig. 4. Using the descriptors, we finally define categorization of more general functions shown in Fig. 5. Note that Fig. 5 shows only an *is-a* hierarchy for readability, because some distinctions are independent from each other. Figs. 4 and 5 are screen-shots of our role-centric ontology editor Hozo (http://www.hozo.jp) [39]. In the Figs. 4 and 5, concepts (classes) are represented as nodes (so-called frames) with slots (right-angled links). The *is-a* relations among concepts (straight links with "is-a". so-called *a-kind-of*, *generic-specific*, *super-sub* relations or *subsumption*) show hieratical categorization. A concept has slots for part concepts (*part-of* relation or *whole-part* relation denoted by a right-angled link with "p/o"), slots for attributes ("a/o") and/or slots for participant concepts (participate-in relation. denoted by "p/i").

In the generalized model shown in Fig. 3, a physical entity (agent (A)) affects a target thing (operand (O)) at a location (L). (The letters such as 'A' are used as identifiers (ID codes) of the descriptors in Figs. 3, 4 and 5). Thus, a state of the operand changes in a time interval (T). The change, i.e., effect (P), is represented as combination of the initial state s_1 at the start time point t_1 and the final state s_2 at the final time point t_2.

The change plays a *goal-oriented role* under a *goal-oriented context*. The *goal-oriented context* consists of *teleological context* (G) and *effect context* as shown in Fig. 4 with a 'part-of' right-angled link. The *teleological context* is related to user's intension as discussed in the following section. The *effect context* specifies the focused area in the model for capturing a role. It is described using sub-types of the elements A, O, L, T and P shown in Figure 4. By specifying the effect context, we can define sub-types of the *goal-oriented role* as shown in Fig. 5 with a straight 'is-a' link.

Note that both Figs. 4 and 5 represent no causal relation. It can be represented in functional models or the way of function achievement, in which the instances of these types form causal and teleological structures.

3.2. Teleological Context

The *teleological context* (G) is categorized into *function context* (GF) (discussed in Section 2) and *unintended context* (GU) shown in Fig. 4 (The code GF denotes that it is a subtype of the *teleological context* 'G' and that 'F' denotes the *function context*). The *function context* represents that the effect (behavior) is 'intended' by a user. With the function context, we can define *general function* as a subtype of the *goal-oriented role* as shown in Fig. 5. As discussed in Section 2, the *function context* is categorized into the *external (user) function context* (GFU) and the *system function context* (GFS).

On the other hand, the *unintended context* is used for unintended phenomena such as faults. The causal process of faults can be represented as a goal-oriented achievement structure for (quasi-)goal. This is used for fault modeling [20]. We shall revisit this issue in Sections 3.5 and 6.2.

3.3. Device and Environmental Functions

The *general function* discussed above has a sub-type; *effect function* as shown in Fig. 5. (Another sub-type; *quasi-function* will be discussed in Section 3.6). It is based on *effect* (the class code PP in Fig. 4) which represents temporal changes of attributes of an external thing other than the agent. Its agent is either device or function (AA).

The *effect function* is further categorized into *device function*, *environmental function* and *system-interface function* according to the descriptors about the location of the operand (L). The *device function* represents changes of entities within the system boundary (class LS). The *environmental function* includes changes outside of the system boundary (class LE), especially, those related to users or user actions. For example, as mentioned in Introduction, an electric fan performs the following two functions; "to move air" as a *device function* and "to cool human body" as an *environmental function*. In the latter, the cool-down effect by wind is on human body and thus outside of the system boundary. The *environmental function* is further categorized into *physical environmental function* and *interpretative function*. The former means physical changes of the system like the cooling function of the fan, while latter sets up one of necessary conditions of human's cognitive interpretation (class OHC in Fig. 4). For example, a clock has the following two functions; "to rotates hands (in the specific and constant rate)" as *device function* and "to inform time." as an *interpretative function*. The latter requires human's cognitive inter-

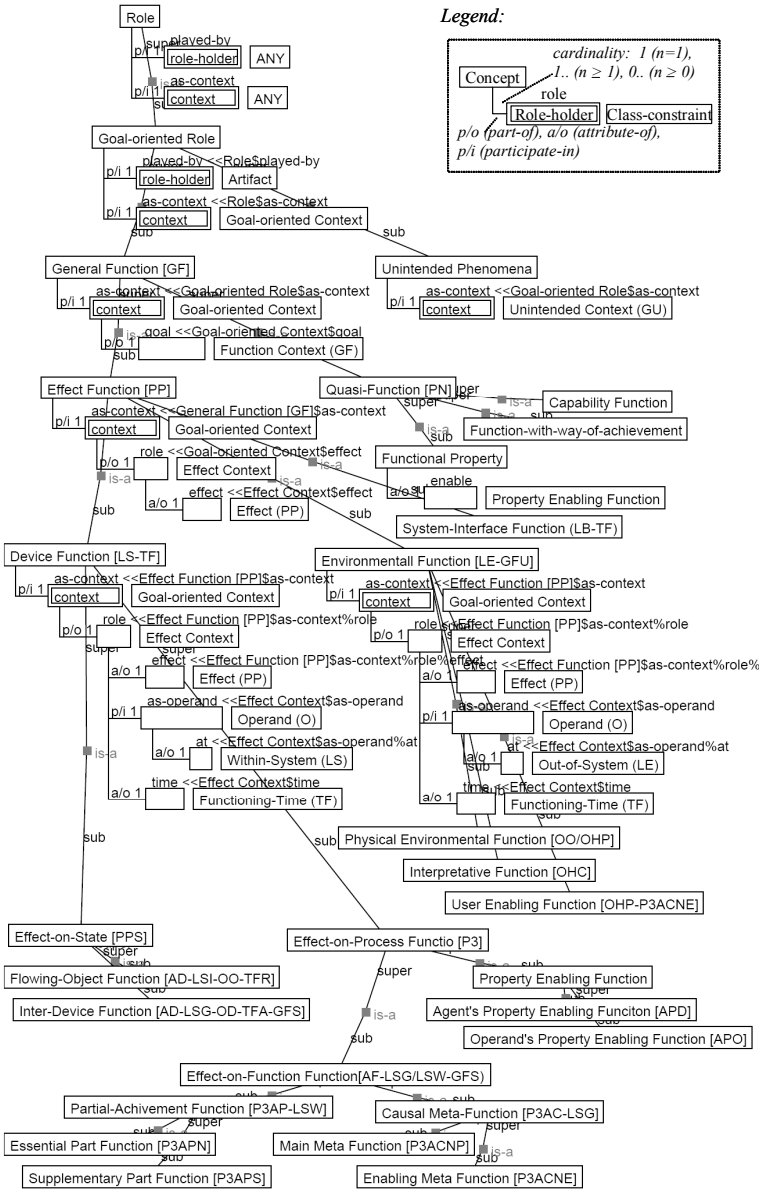


Figure 5. Categories of functions using the descriptors in Fig. 4

pretation. Lastly, the *system interface function* represents effects only at the system boundary (class LB) such as "import something (from the external of the system)".

Chandrasekaran and Josephson discuss a kind of function similar to the environmental function called environment function as effect on environment (the surrounding world of the device) [4]. Although we define function as a role of behavior under a specific context, in [4], the concept of role is used as natural (without human's intention) effects on environment and function is defined as "role + intention". Some researchers distinguish purpose from function (e.g., [14][23]), where the purpose represents human-intended goal in the similar sense to this *environmental function* or *interpretative function*. Rosenman & Gero investigate *purpose* in *socio-cultural environment* [35]. The situated FBS framework treats change of requirements [10]. The affordance-based design has been investigated in [26].

3.4. Effect on State or Process

The *device function* is categorized into *effect-on-state function* and *effect-on-process function* as shown in Fig. 5. The *effect-on-state function* refers to changes of physical attributes of a target thing (class PPS in Fig. 4). It has a sub-type, *flowing-object* function, which corresponds to the *base-function* in Section 2. It refers to flowing-objects (OO) as an operand, a device as an agent (AD), and 'device-oriented behavior' discussed in Section 2 as change of the operand. It is change of physical states (PPSAC) at device's input and output ports (LSI) in the time-interval related to operand's flow from the input port to the output port (TFR). Another sub-type, *inter-device function* which refers to changes of another device. Its example is a rod's function "to push cam". The cam is another device, which is not considered as objects flowing through the rod.

The *effect-on-process function* is based on effect on a process or change (P3). Behavior as the basis of *device function* can be regarded as a kind of a process. It has a sub-type, *effect-on-function function*, where a function plays a specific role for another function. The *effect-on-function function* is categorized further into *causal-meta function* and *partial-achievement function.* The former represents causal relationship between the functions of the sibling devices at the same level (P3AC-LSG). Similar assisting function has been categorized [14]. We also identified "meta-function" such as ToDrive and ToEnable, which is a collaborative role played by a base-function for another base-function [19].

The latter is performed by a *method function* for a *goal function* in the *is-achieved-by* relation in the function decomposition (P3AP-LSW). A goal function of a system can be achieved by a series of method functions of components. Function defined in [4] is related with this relationship. The partial function [14] is similar to this.

The method functions are categorized into *essential part functions* and the *supplementary part functions* according to whether the contribution is mandatory or not (P3APN/P3APS). Distinction between primary function and secondary function [33] is similar to this.

3.5. Negative Goal and Kinds of Time Interval

A part of subclasses of effect (P) and time (T) shown in Fig. 4 can be additional descriptors for the functions discussed thus far. The subtypes of the *state achievement* descriptor (PPSA) represent causal patterns of achievement for goals as Function Types such as ToMake and ToMaintain [19]. Garbacz defines participation functions according to categories of perdurants (i.e., process) in an ontological meta-theory [11].

On the other hand, *negation-of-state* (PPSP) and *negation-of-process* (P3P) have goals which prevent that a specific state (or a process) happens (we call *negative function* as an antonym of *positive function*). The following two functions of a paperweight have a negative goal and a normal goal, respectively.

"A paperweight prevents such a possible process that a piece of paper moves." [Negation-of-process (P3P)]

"A paperweight exerts vertical force on a piece of paper." [Change-state1-to-state2 (PPSAC)]

The relationship among these two functions and a related unintended phenomenon is illustrated in Figure 6. The effect model (a) (in the left) shows the latter function "to exert force on the paper". The model (c) (in the right) shows that change-motion of the piece of the paper by wind is interpreted as an unintended phenomenon "move paper (deviate from the proper position)" under this teleological context as discussed in Section 3.2. The model (b) (in the center) shows the former function "to prevent the paper's moving process". Precisely, this preventing function is performed by the exerting function as shown Fig. 6(b). The exerting function affects (prevents) the initiation of the moving process. Thus, this function is an *effect-on-process function* mentioned in Section 3.4. This model is a simplified model which omits force-level behavior for the paper's movement, the helping-human's-writing function, and so on. We are currently investigating such a full model.

Each of these three models is captured from different 'cap-

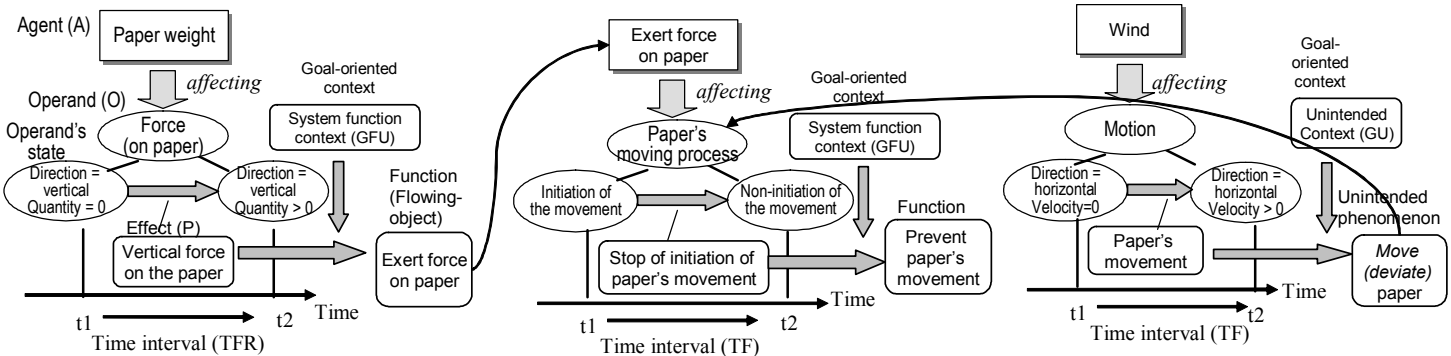


Figure 6. A model of a paper weight's two functions (a) and (b) with an unintended phenomenon (c).

turing perspectives'. The models shown in Figures 6(a), (b) and (c) capture "output of force", "preventing function" and "prevented (unintended) phenomena", respectively. A model-author might choose one of those perspectives unconsciously. The clear discrimination among those perspectives helps model-authors describe consistent models as discussed in Section 5.

The effects are represented as changes of values of attributes in a time interval. The same effect can imply different meanings for different kinds of the time intervals as subtypes of descriptors of time 'T'. For example, an increasing-temperature function of a heater can imply;

"A heater increases the temperature of the air at a specific location in a room." [Absolute functioning time function (TFA)+LSG]

"A heater increases the temperature of the air at the output port to higher than at the input port." [Flowing-object functioning time function (TFR+LSI)]

The first one refers to the changes of operand's states at a specific absolute location in a time interval. The second one refers to the change during an entity which flows from the input port to the output port, which are specified relatively from the device. These two are within *functioning time* (TF).

On the other hand, the d*esigning-Time* (TD) descriptor for the time interval of the effects is totally different from them in the *functioning time*. For example, one might describe a function of diamond powders in a cutting machine as follows;

"The diamond powders increase the frictional coefficient of the cutting blade." [Designing-Time (TD)]

When the cutting machine is functioning, the frictional resistance of the cutting blade is kept high. The increase refers not to change in the *functioning time* (TF) but to comparison with the design case without the diamond powders in the (re-) design processes in the *designing-time* (TD). We consider this is not function but a *quasi-function*. In the deployment [19], however, many engineers descried like this. It shows a justification of needs of these concepts about time-interval as kinds of capturing perspectives.

3.6. Quasi-Functions

We recognize the following kinds of quasi-functions. Although the authors do not consider them as kinds of function, it is found that a quasi-function is confused with a function. A *function-with-way-of- achievement* implies a specific "way of function achievement" as well as a function. For example, "to weld objects" as a function of a manufacturing facility implies not only "what to achieve", say, to join them, but also a certain way ("how") to achieve the goal, the objects are fused. In fact, the same goal can be achieved in different ways (e.g., using bolts and nuts) without fusion. Other examples include washing, shearing, adhering (e.g., glue adheres A to B), linking [13] and "transportation by sea". Because meaning of this type of function is impure, we regard this quasi-function. In our functional ontology, the "way of function achievement" is separately conceptualized [19]. It is similar to "means" in [3][27].

A *functional property*[1] represents that an artifact (usually material) has a specific attribute-value which directly causes a function. This is found in materials science domain where a material whose function is dependent on its electronic, optical or magnetic property is called functional material [8]. This is (usually implicitly) based on the *property enabling function* as discussed below.

The *property enabling (quasi-)function* is based on effect on behavior (effect) (P3ACNE) performed by a physical *property* (AP). What we mean by a property here is a conceptualization of "having (a range of) a value of an attribute". Because a property inheres in a physical entity, we regard this quasi-function. An example is;

"The high conductive property of a conducting wire enables its electric-conducting behavior."
[Property enabling (quasi-)function]

This high-conductive property is one of the necessary conditions for performing the conducting behavior. Such relationship exists in all realization of behavior. If the property is a key factor for realization of the behavior, it might be captured as this quasi-function. Moreover, enabling behavior implies enabling performance of function. Especially, in this case, there is direct relationship between the conductive property and the conducting function. Thus, the conductive property is called *functional property* as discussed above. Such functional property provides a part of justification for device's *capacity to perform a function* discussed in Section 2.

As well as a *property of agent* of function (APD) such as the high-conductive property discussed above, a *property of an operand* (APO) can enable behavior of a device like as follows;

"The combustible property of oil enables the burning behavior of a boiler."
[Operand's property enabling (quasi-)function]

In this case, the burning behavior plays a heat-generation function. Especially, in case of human actions, we can say "a property of a *tool* enables human actions" as follows;

"The sharp-edge of a knife enables human's cutting action." [User enabling (quasi-)function]

This is defined as *user enabling function* as a sub-type of *environmental function*. This is similar to the affordance-based manner of artifacts in [26].

A *capability function* represents that an entity can perform an activity which is not effect on others. For example, people say as follows;

"Humans have walking function" [Capability function]

The *property enabling function* and *capacity to perform a function* are similar, but they are based on effect on other external entities. Lastly, functions with designing-time (TD) discussed in Section 3.5 are quasi-functions as well.

4. OTHER DISTINCTIONS OF FUNCTIONS

The types of function discussed in Section 3 are categorized according to their elements of the effect model shown in Fig. 3. Other distinctions remain as follows.

Locality: A function of a component can be different according to its reference points. We can categorize them into the following major types. The *flowing-object function* shown in Fig. 5 refers to local behaviors (LSI in Fig. 4) in the device (we call *local function*), though it is dependent on context. On the other hand, we call functions that refer to non-local points (LSG, LSW or LE) *conjunct functions*. For example, the heat exchanger discussed in Section 2 can perform conjunct functions such as controlling temperature of a room or preventing

[1] The term "functional" here is intended to represent neither mathematical dependence relation nor attributes of function but function-oriented property. Functional property is used as an antonym of mechanical or structural property.

engine's overheating. They cannot be exhaustively enumerated in nature. In order to realize composability of device models, we have to distinguish these two types.

Essentiality: An artifact can perform (at least) a function that is intended by a designer (we call *essential function*). The essential function provides artifact's identity. On the other hand, a user can use a device differently from the use intended by the designer. In such a case, we recognize the device performs an incidental function (we call *accidental function*) induced by the use. For example, a screw driver performs a screwing function as its essential function. A user can use it for hitting (exerting linear force on) a nail as an accidental function. The screwing function could be performed by a key as its accidental function.

Process v.s. property: People might say that a device 'performs' a function and/or 'has' a function. The former function to be performed is, we believe, a role (or kind) of process of the device. The latter function to be had seems to be a kind of property (or characteristics) of a device. In this article, we discuss the former mainly and treat the latter as *a capacity to perform the function* as discussed in Section 2. Hubka and Eder define functions as a property as follows; "The function is a property of the technical system, and describes its ability to fulfill a purpose, namely to convert an input measure into a required output measure under precisely given condition" [14]. In philosophy, a function is typically a special feature of artifacts (especially of biological organs) [34]. In [16], function of a biological organ is defined as "the disposition of a certain entity reliably to act in such a way as to achieve a goal".

5. MODELING BAESD ON PRESCRIPTIVE DEFINITIONS

One of the fundamental usages of the upper-level categories of function discussed thus far is to help model-authors capture functions consistently as a *prescriptive definition*. As discussed in the Introduction, without guidelines, engineers tend to describe functional models in an ad hoc way. For consistent modeling, the modeler firstly selects acceptable type(s) of function in the reference ontology. The specifications of the type(s) of the function defined in the reference ontology provide model-authors hints and restrictions for capturing functions. For example, if the *flowing-object function* is selected as the acceptable type in a model, the modeler should focus their attention on the effect by a device (AD), on the local input and output of the device (LSI), and on the change of the object flowing through the devices (OO) in the functioning time (TFR) (The codes show elements defined in Fig. 4 for definition of the flowing-object function). Such specification provides prescriptive guidelines with modelers. Needless to say, such consistency and uniformity of the models greatly help knowledge sharing among engineers.

Even if some types were selected as acceptable types, it would be beneficial for understanding and interoperability of the knowledge that the relationship and the differences among the written functions would become clear.

6. ONTOLOGY MAPPING FOR INTEROPERABILITY

One of the other important usages of the upper-level categories of function is to be used as a *reference ontology* for interoperability between different kinds of knowledge. By reference ontology, we here mean an ontology referred to for categorizing

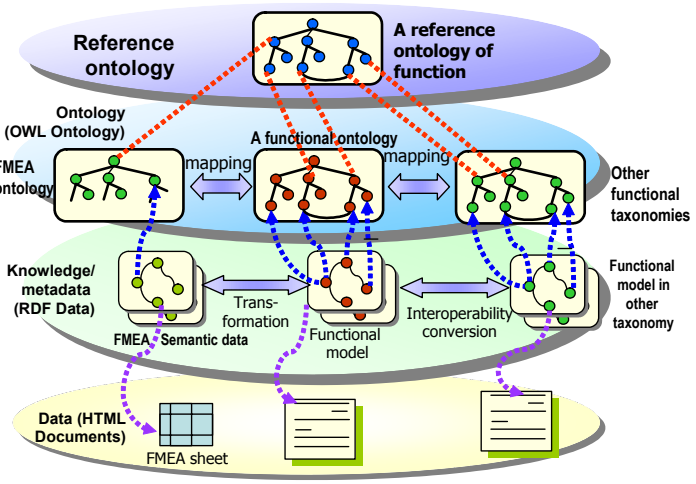


Figure 7. Knowledge (metadata) interoperability using ontology mappings based on a reference ontology of function

existing definitions of function and mapping them (in comparison with "reference for system design" such as the ISO's OSI network reference model). Using a reference ontology, interoperability with other kinds of knowledge can be realized based on ontology mapping as shown in Figure 7. Each knowledge type has own ontology. Interoperability between two knowledge types can be realized using ontology-mapping between their ontologies via the reference ontology. Ideally, each concept in the target ontology can be categorized in the reference ontology. Such categorization in an upper-level ontology facilitate the ontology mapping. Such effect of an upper-level ontology on ontology mapping or alignment is widely recognized in the literature such as in [31].

The following sub-sections show two examples of such ontology-based interoperability. Section 6.1 discusses interoperability between different functional taxonomies. We give here an overview and effects of the reference ontology in such interoperability. The detail of the mapping is discussed in [32]. Section 6.2 discusses integration of fault knowledge with functional knowledge. This is a generalized account of the framework discussed in [20].

6.1. Mapping between functional taxonomies

In order to share functional knowledge in an organization, one of the important ways is to use taxonomy of functions, whereby functions of components in functional models are described in terms defined in a taxonomy. In fact, some taxonomies of generic functions have been proposed in [11][13][33][40]. When different taxonomies are used in functional models, the interoperability among them becomes problematic. We need some kind of *mappings* between terms in the different taxonomies. Between the taxonomies, however, there are not only terminological differences but also implicit ontological differences.

The reference ontology proposed in this paper can be used for revealing such ontological differences in order to help mapping-authors determine which function terms represent similar notion of function. Ideally, a function term in a functional taxonomy is categorized into an upper class in the reference ontology. The term can be mapped to other term(s) in other functional taxonomy according to the categories in the reference ontology which the terms are categorized to.

Such reference-ontology based mapping between func-

tional taxonomies is discussed in [32], though the detailed contents of the reference ontology of function is discussed only in this article. The paper [32] uses the Reconciled Functional Basis (hereafter FB) [13] and the revised version of our functional concept ontology (hereafter FBRL) [19] as an example of the mapping. For example, both the term "couple" in FB and the term "combine" in FBRL are categorized into the *flowing-object function* class in the reference ontology. Then, both the terms are directly mapped to each other. It is just terminological difference. On the other hand, the functional term "link" in FB implies not only "to couple flows together" [13] as the change at input and output but also "by means of an intermediary flow" [13] as how to realize it. According to the definition and its example, the "link" in FB seems to be categorized into the *function-with-way-of-achievement* category. Thus, it cannot be fully mapped to "combine" in FBRL which implies "to bring two operands into an operand" as the change at input and output, which corresponds to only the former part of the meaning of "link". This is not a terminological but an ontological difference, because "the change in the target object" and "how to realize the change" are ontologically different. The difference of the classes; *flowing-object function* and *function-with-way-of-achievement* clarifies the ontological difference.

Another example of ontological mismatch is observed at "import" and "export" in FB. These concepts are categorized into *system-interface function* in the reference ontology as discussed in Section 3.3. FBRL does not have such functional concepts, which can be expressed by input or output from the boundary of a model.

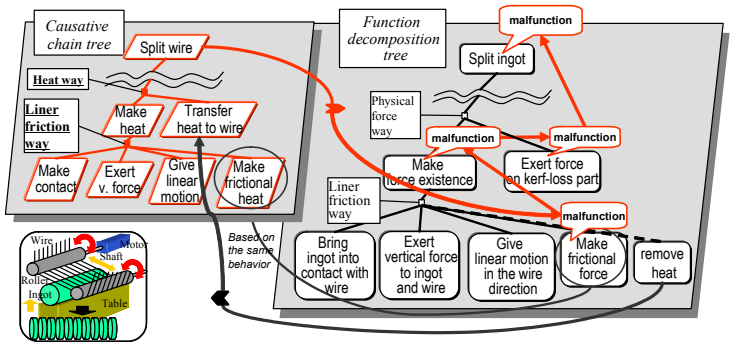
The concrete mappings are currently under investigation. Please refer to another paper for the details of the current mappings and the translation of the functional models [32]. Although the mapping is hand-written by the authors (not by engineers) at this moment, the model transformation can be done semi-automatically.

Thus, the reference ontology of functions can be used for clarifying such ontological differences between the functional taxonomies and for enabling translation between them. Note that the function categories in the reference ontology are types of ontological definitions of functions and then are super-types of the function classes in the taxonomy. Thus, a set of the function terms in a functional taxonomy is *not* sub-set of the reference ontology. It is not our goal to build such an ontology that *subsumes* (includes) all the functions in the existing taxonomies as a kind of a super all-inclusive set.

6.2. Integration of fault knowledge in FMEA

FMEA (Fault Modes and Effects Analysis) is one of the typical forms of fault knowledge in industry. It includes possible fault modes of components, their causes, and their propagation of effects in the system, which are usually unintended by designers (and users). As discussed in Section 3, we can categorize such causal processes as processes of *unintended phenomena* under 'unintended context'.

Thus, we can integrate such unintended behaviors such as faults into a functional model in a generic effect model. Figure 8 shows a model of a possible fault mode and its effects of a wire saw, which slices semiconductor ingots using a moving wire. Each element in the model is categorized into combination of the classes in the reference ontology. The tree on the right plain mainly consists of usual functions ("Flowing-Object



Component	Function	Failure mode	Cause	Effects	Action
Wire	To make frictional force	Malfunction	To make frictional heat -> To split wire	Malfunction of "to split ingot"	To remove heat

Figure 8. Integrated model of function and unintended phenomena of a wire-saw and a FMEA sheet translated from the integrated model.

Function"). The left tree shows a possible causative chain (process) of malfunction of the wire saw, in which frictional heat causes a snap of the moving wire. This process consists of "unintended phenomena + flowing-object function". Although it cannot be called as a function due to its unintendedness, we can use the same set of functional concepts (e.g., "to split wire"). An arrow from the left tree to the right tree represents the effects that the snap of wire causes the malfunction of the wire. It can be categorized by "unintended phenomena + effect-on-process + negation-of-process" as discussed in Section 3.5. Our previous paper [20] reports a modeling framework of such an integrated model, which is a special (limited) case of the generic model in this paper.

An ontology mapping between the extended functional ontology and an FMEA ontology has been described. It shows the correspondences between concepts in our functional ontology and those of the FMEA ontology. We had some difficulties in making the mappings due to ambiguous meanings and uses in the concepts of FMEA. For example, "failure mode" in FMEA is defined as "the manner by which a failure is observed" [30]. In FMEA documents in practice, however, "inefficient", "vibration" and "fracture" are found. These seem to represent "undesirable status of function", "cause of fault" and "symptom", respectively [20]. We managed to describe a mapping relation between "failure mode" in FMEA and *function defect* in our ontology (we changed the previous mapping in [20] through the experiment with JAXA discussed below). Using the reference ontology, we can categorize the concept "failure mode" in FMEA into "unintended phenomena + Nagative function". The reference ontology helps such complicated ontology mappings.

Based on the mapping knowledge, a transformation system can generate FMEA documents from an integrated model [20]. The table at the bottom of Figure 8 shows a FMEA sheet which is automatically translated from the integrated model.

This framework has been evaluated in collaborative research with Japan Aerospace Exploration Agency (JAXA). We

described an integrated model of the electrical power subsystem of a satellite, which includes its functional structure and possible faults. Then, we inputted the integrated model into the transformation system and then got auto-generated FMEA sheets. Lastly, we compared the auto-generated FMEA with the FMEA sheets that were previously described by domain experts of JAXA directly (referred as the original FMEA hereafter). As the result of a comparison between the auto-generated FMEA and the original FMEA, we found the auto-generated one includes some extra fault-modes and possible effects than the original one. Those fault modes and possible effects were implicit in the original. Note that, in the real satellite system, those implicit fault modes (even if happened) and effects never cause real serious effects. Even so, according to the aim of the FMEA activity, exhaustive enumeration of fault modes and effects is expected. In summary, the proposed transformation method can improve the exhaustiveness of fault knowledge. The proposed method aims *not* at auto-generation (reasoning) of fault phenomena *but* at helping knowledge-authors describe their knowledge exhaustively. Integration of description of functional knowledge and fault knowledge gives engineers good stimuli for enumerate possible faults and effects.

We have developed an integrated search system based on this ontology mapping as an extension of the *Funnotation* search system reported in [17]. It enables the designers to search web-documents using metadata (semantic annotations to the documents) based on the Semantic Web technology. The designer can search both document types of functional structures and FMEA sheets by a single query. It is enabled by the ontology mapping discussed above. Moreover, the system can transform the functional structure model (metadata) into FMEA sheets automatically using the ontology mapping.

7. RELATED WORK AND DISCUSSION

Many surveys or categorizations of definitions of functions have been reported to date in engineering design [4][9][38] and philosophy [34]. We do not aim at such exhaustive categorization of the existing definitions in the literature. Rather, we aim at clarifying possible perceptions (viewpoints) of functions on an explicit (basic) effect model. Using the types of perceptions, as a part of the results, we identify some types that are similar to the existing definitions in the literature as discussed in Section 3. Such types have clear meaning based on the generic model and clear discrimination with other types. Moreover, the reference ontology of function provides *neither* a super-set (logical sum) *nor* a standard set of the concrete functions of devices/components. Rather, it provides generic upper-level categories to categorize them.

The layered structure of the ontologies shown in Fig. 7 is commonly used in Ontological Engineering such as PhysSys [2] and FBSO ontology [24]. Our reference ontology of function is at the middle-level layer in such general layered structure of ontologies. In general, an ontology at the most top-level layer is called “top-level ontology” or “upper ontology”, which includes meta, abstract, and philosophical concepts such as DOLCE [12]. An analysis of some upper-level ontologies has been done for the manufacturing domain [36]. PSL treats general (discrete) “process” such as manufacturing process [15]. Our reference ontology of function is the level lower than those upper-levels and includes several function concepts that are general in engineering domain. A methodology for creating engineering ontologies reported in [1] includes reuse of existing taxonomies, test for application, and refinement of the integrated taxonomy.

The ontology-based integration and interoperability among design knowledge have been investigated such as PACT [6] and KIEF [43]. They mainly focus on generic interoperable mechanism among agents and/or engineering tools. Product data exchange based on ontology has been proposed in [5]. An ontology-based design knowledge modeling is discussed in [24]. We aim at generic and richer ontology of function and clear conceptualization of related types. DAEDALUS knowledge engineering framework [22] is a framework for knowledge sharing between design and diagnostic tasks. We use rich functional ontologies and ontology mapping.

Automatic generation of FMEA documents from behavior or functional models such as FMAG [41] and Advanced FMEA [37] has been investigated. In addition to fault modes and effects in those models, our integrated model includes a detailed causative process of faults as shown in Section 6.2.

TRIZ (TIPS) theory provides some patterns (or strategies) for inventions based on the contradiction between two physical quantities [40]. We did not concentrate on design strategies but on modeling schema. TRIZ theory also concentrates on physical principles (effects), although we established a clear relationship between physical principles and functional structures.

In the research field of ontological engineering, much research has been carried out on automatic ‘mapping discovery’ which is to find similarities between two ontologies and determine which concepts represent similar notions [31] based on lexical information, semantics of relationships, and/or a set of shared instances of classes. It is pointed out that such (semi-) automatic finding can be facilitated by a common grounding such as a shared upper ontology [31]. Our reference ontology, we think, can have the same effect on the automatic mapping discovery for functional taxonomies.

Interoperability among diverse definitions of function is important, especially in the Semantic Web. For example, information integration of product data in Semantic Web is required and realized by ontology mapping [25]. A functional annotation schema for the Semantic Web is proposed based on the functional basis as taxonomy [21]. Our ontological work here aims at clarifying a viewpoint for capturing functions for such the interoperability.

8. CONCLUSIONS

This paper proposed a categorization of a variety of functions using a reference ontology of functions we built. The proposed categorization is based on a generic model of the standard input-output model. This paper overviewed two usages of the categorization as a reference ontology as well, in which it can be used for mapping concepts or terms among different knowledge forms such as FMEA and different functional taxonomies.

ACKNOWLEDGMENTS

The authors thank Yusuke Koji, Masanori Ookubo, Munehiko Sasajima, Kouji Kozaki and Masataka Takeuchi for their contribution. Special thanks go to Mr. Yoshio Tsutsui and Dr. Yoshikiyo Kato, Japan Aerospace Exploration Agency (JAXA), for their cooperation in the application of our methods in a space system.

REFERENCES

[1] Ahmed, S., Kim, S., Wallace, K. M., 2005, "A Methodology for Creating Ontologies for Engineering Design". In *Proc. of ASME IDETC 2005 DTM*, DETC2005-84729.

[2] Borst, P. Akkermans, H., and Top, J, 1997, "Engineering Ontologies", *Human-Computer Studies*, **46**(2/3), pp. 365-406.

[3] Bracewell, R.H., Wallace, K.M., 2001, "Designing a Representation to Support Function-Means based Synthesis of Mechanical Design Solutions", In *Proc of ICED 01*.

[4] Chandrasekaran, B, Josephson, J.R., 2000, "Function in Device Representation", *Engineering with Computers*, **16**(3/4), 162-177.

[5] Christel D. and Parisa C., 2002, "Product Data Exchange Using Ongologies", In *Proc. of Artificial Intelligence in Design* (AID2002), Cambridge, UK, pp. 617-637.

[6] Cutkosky, M.R., et al., 1993, "PACT: An Experiment in Integrating Concurrent Engineering Systems", *Computer*, January:28-37.

[7] De Kleer J, Brown J.S., 1984, "A Qualitative Physics based on Confluences", *Artificial Intelligence*, 24:7-83.

[8] EPSRC (Engineering and Physical Sciences Research Council) (2005) http://www.epsrc.ac.uk/ResearchFunding/Programmes/Materials/ResearchPortfolio/EngineeringFunctionalMaterials.htm

[9] Far, B. H., Elamy, A. H., 2005, "Functional reasoning theories: Problems and perspectives", *AI EDAM*, 19, pp. 75-88

[10] Gero, J.S., Kannengiesser, U., 2002, "The Situated Function-Behaviour-Structure Framework", In *Proc. of Artificial Intelligence in Design '02*, pp. 89-104.

[11] Garbacz, P., 2005, "Towards a Standard Taxonomy of Artifact Functions", In *Proc. of the First Workshop FOMI 2005 - Formal Ontologies Meet Industry*, CD-ROM.

[12] Guarino N., 1997, "Some Ontological Principles for A Unified Top-Level Ontology", *Proc. of AAAI Spring Symposium on Ontological Engineering*.

[13] Hirtz, J., Stone, R.B., McAdams, D.A., Szykman, S., Wood, K.L., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", *Research in Engineering Design*, **13**, pp. 65-82.

[14] Hubka, V., Eder, W.E., 1988, *Theory of Technical Systems*, Berlin: Springer-Verlag.

[15] ISO TC184/SC4/JWG8, 2003, Process Specification Language, http://www.tc184sc4.org/SC4_Open/SC4_Work_Products_Documents/PSL_(18629)/

[16] Johansson, I., Smith, B., Munn, K., Tsikolia, N., Elsner, K., Ernst, D., and Siebert, D., 2005, "Functional Anatomy: A Taxonomic Proposal", *Acta Biotheoretica*, 53(3), pp. 153-66

[17] Kitamura, Y., Washio, N., Koji, Y., Sasajima, M., Takafuji, S., Mizoguchi, R., 2006, "An Ontology-Based Annotation Framework for Representing the Functionality of Engineering Devices, In *Proc. of* DTM of *ASME IDETC/CIE 2006*, DETC2006-99131.

[18] Kitamura, Y, and Mizoguchi, R., 2007, "A Device-oriented Definition of Functions of Artifacts and Their Perspectives", Manuscript for *the 15th Altenberg Workshop in Theoretical Biology "Comparative Philosophy of Technical Artefacts and Biological Organisms"*, MIT Press, to appear.

[19] Kitamura, Y., Koji Y., and Mizoguchi, R., 2006, "An ontological model of device function: industrial deployment and lessons learned", *Applied Ontology*, **1**(3-4), pp. 237-262.

[20] Koji, Y., Kitamura, Y., Mizoguchi, R., 2005, "Ontology-based Transformation from an Extended Functional Model to FMEA", In *Proc. of ICED 05,* 264.81.

[21] Kopena, J. B., Regli, W. C., 2003, "Functional Modeling of Engineering Designs for the Semantic Web", *IEEE Data Engineering Bulletin*, IEEE Computer Society, **26**(4), pp. 55-62.

[22] Lee, B. H., 2001, "Using FMEA models and ontologies to build diagnostic models", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**, pp. 281-293.

[23] Lind, M., 1994, "Modeling goals and functions of complex industrial plants", *Applied Artificial Intelligence* **8**, pp. 259-283.

[24] Li, Z., Anderson, D. C., Ramani, K., 2005, "Ontology-based Design Knowledge Modeling for Product Retrieval", In *Proc. of ICED 2005*, 462.1.

[25] Maier, A., Schnurr, H. P., Sure, Y., 2003, "Ontology-based Information Integration in the Automotive Industry", In *Proc. of ISWC 2003*, pp. 897-912.

[26] Maier, J. R. A. & Fadel, G. M., 2003, "Affordance-based methods for design", *Proc. of DETC 03*, DTM-48673

[27] Malmqvist, J., 1997, "Improved Function-means Trees by Inclusion of Design History Information", *Journal of Engineering Design* **8**(2), pp. 107-117.

[28] Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gengami, A., and Guarino, N., 2004, "Social Roles and their Descriptions", In Proc. of the *9th Int'l Conf. on the Principles of Knowledge Representation and Reasoning (KR2004)*, 267–277.

[29] Miles, L. D., 1961, *Techniques of value analysis and engineering*. McGraw-hill.

[30] Military Standard, 1980, Procedures for Performing a Failure Mode, Effects and Criticality Analysis, MIL-STD-1629A.

[31] Noy, N., 2004, "Semantic integration: A survey of ontology-based approaches". *ACM SIGMOD Record archive*, 33(4).

[32] Ookubo, M., Koji, Y., Sasajima, M., Kitamura, Y., Mizoguchi, R., 2007, "Towards Interoperability between Functional Taxonomies using an Ontollogy-based Mapping", In *Proc. of ICED 2007*, to appear.

[33] Pahl, G., and Beitz, W., 1998, *Engineering Design - a Systematic Approach*. The Design Council.

[34] Perlman M, 2004, "*The Modern Philosophical Resurrection of Teleology*". The Monist 87(1):3-51.

[35] Rosenman, M.A. & Gero, J. S., 1998, "Purpose and function in design: from the socio-cultural to the techno-physical", *Design Studies*, *19*, 161-186.

[36] Schlenoff, C., Denno, P., Ivester, R., Libes, D., Szykman, S. 2000, "An Analysis and Approach to Using Existing Ontological Systems for Applications in Manufacturing", *AI EDAM*, 14, 257-270.

[37] Steven, K., Peder, F., and Ishii, K., 1999, "Advanced Failure Modes and Effects Analysis of Complex Processes", *Proc. of the ASME Design for Manufacturing Conference*.

[38] Stone, R. B., Chakrabarti, A. (eds.), 2005, "Special Issues: Engineering applications of representations of function," *AI EDAM*, **19**(2 and 3).

[39] Sunagawa, E., Kozaki, K., Kitamura, Y., Mizoguchi, R., 2006, "Role organization model in Hozo", In *Proc. of EKAW 2006*, LNCS 4248, Springer, pp. 67-81.

[40] Sushkov, V.V., Mars, N.J.I., Wognum, P.M., 1995, "Introduction to TIPS: a Theory for Creative Design", *Artificial Intelligence in Engineering,* **9**(3), pp. 177-189.

[41] Teoh, P.C. and Case, K., 2004, "Modelling and Reasoning for Failure Modes and Effects Analysis Generation"', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **218**, pp 289-300.

[42] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., 1996, "Supporting conceptual design based on the function-behavior-state modeler" *AI EDAM* **10**, pp. 275-288.

[43] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T., 2004, "Physical Concept Ontology for the Knowledge Intensive Engineering Framework", *Advanced Engineering Informatics,* **18**(2), pp. 95-113.

