In Proc. of the 14th International Conference on Knowledge Engineering and Knowledge Management EKAW 2004, to appear.

Ontology-based Functional-Knowledge Modeling Methodology and its Deployment

Yoshinobu Kitamura and Riichiro Mizoguchi

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

Abstract. Functionality is one of the key concepts in understanding an artifact and in engineering domain knowledge. Although the importance of sharing of engineering knowledge in industry has been widely recognized, from our experience with collaborative research with a production company, industrial engineers have had difficulty in sharing engineering knowledge including functionality. To promote the sharing of the engineering knowledge from the viewpoint of functionality, we have established an ontology-based knowledge modeling methodology for functional knowledge, which has been successfully deployed in a production company. It consists of two ontologies to capture functionality and the specifications for modeling processes. This paper summarizes these ontologies and its deployment, and discusses the modeling process based on the ontologies, which includes detailed modeling steps, types of functional knowledge, and ontological guidelines.

1. Introduction

Understanding an artifact is a major part of domain knowledge. Functionality is one of the key concepts in understanding an artifact. While there is no common understanding of what a function is [1-4], people share the idea that functional knowledge is tightly related to design intention. In contrast to objective data about an artifact such as dimension, shape and structure, recognition of functionality is dependent on systems, environments or situations in which they are embedded. A function of a device explains what users can get using it in an environment (effects or worth of the artifact). A function of a component embedded in a system explains why the component exists in the system and how it contributes to achieving the system's whole-function. In the problem of design and manufacturing, such functional knowledge represents designer's intention (so-called design rationale (DR)). It plays a crucial role in engineering tasks such as designing and trouble shooting by engineers [1-5] as well as understanding artifacts by users.

The importance of the knowledge management (KM) of engineering knowledge in industry is widely recognized. The recent CAD systems and computer network technologies enable engineers to share *the objective data* of an artifact such as shape so-called Product Data Management (PDM). The current KM technology relies mainly on searching documents by keywords. From our experience with collaborative research with a production company, however, industrial engineers have had difficulty in sharing the engineering knowledge among them for long years. They have been regularly writing various kinds of technical reports for each of the jobs such as design review and maintenance. Such documents include real "know-how" in order to keep qualities and avoid troubles. Nevertheless, few of them are retrieved (and reused) by other engineers using the search technologies, because many of these documents are specific to each product from own viewpoint of each engineer. One of its reasons is the lack of semantic constraints (or guidelines) on document contents. We argue that ontologies of functionality can provide semantic constraints/guidelines on knowledge-contents as we will discuss its needs in the next section.

To promote the sharing of the engineering knowledge of artifacts from the viewpoint of functionality, our goal here is to establish an ontology-based knowledge modeling methodology for functional knowledge. We have developed two ontologies to capture functionality, i.e., the extended device ontology [6] for capturing the target world and the functional concept ontology [7] for rich generic functions of components. In addition to these ontologies, *specification on the knowledge-modeling process* plays a crucial role in the practice of knowledge management. It includes steps for knowledge authoring, the types of functional knowledge to be modeled in each step, and ontological guidelines. The modeling methodology has been successfully deployed in a production company [8]. In the deployment, it has been understood that such specification is one of its success factors.

This paper discusses the ontology-based modeling methodology for functional engineering knowledge. We overview the ontologies and its deployment in industry as a success story of Ontological Engineering. In Ontological Engineering research, how to use ontologies in real situations in industry is an important issue as well as theory, methodology and tools. The main topic of this paper is the specification on the modeling processes and guidelines based on the ontologies. The specification has been used in the deployment but has not been reported yet. Although we have reported the contents of ontologies in [6,7] and the deployment in [8], we summarize them from the viewpoint of knowledge modeling.

This paper is organized as follows. The next section discusses the needs of ontologies for functionality. Section 3 provides an overview of the ontologies for capturing functionalities. Section 4 discusses the modeling process based on these ontologies. Section 5 presents its successful deployment with our analysis of the success factors. Section 6 discusses related work, limitations, and application domains for our ontologies. Section 7 concludes the paper.

2. Needs of Ontologies for Functionality

A great deal of work on domain ontologies in the engineering domain has been done [9-11]. These, however, have mainly been concerned not with teleological functionality but objective structures and behaviors. On the other hand, although a great deal of research on the functionality of engineering products has been conducted in engineering design research [3,5,12,13], functional representation research [1,2,4,15-21], and value engineering [22], there have been few ontological considerations [4,23,24].

We think there is a considerable gap between such theoretical research and practice in industry. Here, we present two examples that demonstrate the difficulty in functional knowledge modeling and then the needs of ontologies. Firstly, functionality in Value Engineering is represented in "verb+noun" style [22] and on the basis of this, one might describe "to weld metals" is a function of a manufacturing machine as a keyword for its document. However, "to weld metals" implies both the metals are joined and their parts are fused. From the viewpoint of functionality in manufacturing, joining is only the goal the designer intends to attain, while the fusion can be regarded as a characteristic of "how to achieve that goal". In fact, the same goal, "to join", can be achieved in different ways (e.g., using nuts & bolts) without the fusion. When a designer looks for different ways to achieve a goal function by specifying the function as a keyword, his/her capturing it as "to join" instead of "to weld" enables him/her to find "nuts & bolts" as a possible alternative to "welding". This example demonstrates the importance of the concept of functionality in reusable functional knowledge.

The well-known systematic design methodology in [5], on the other hand, includes hierarchical structures of functionality based on input-output relations (so-called functional decomposition). However, this is not easy to describe such functional models. For the same welding machine, one might describe "to put objects together", "to make an arc", and "to leave them" as sub-functions (decomposed micro-functions) of the goal function "to join". These sub-functions certainly describe decomposition of the input-output relation. However, there is an implicit intermediate function "to heat objects" between "to make an arc" and the goal function. In fact, "to heat objects" can be achieved by "to make current flow" instead of "to make an arc". This second example demonstrates the importance of practical specifications for functional decomposition, in addition to standard specifications as decomposition of input-output.

These suggest the necessity for practical specifications for the content of functional knowledge and of how to describe it. The former can be specified as ontologies, i.e., "explicit specifications of conceptualization" [25]. Ontologies can provide fundamental concepts for capturing the target world in a consistent way and a vocabulary to describe the knowledge. The latter means specifications for modeling processes, which include steps for knowledge authoring and ontological guidelines. Ontologies specify the results of knowledge modeling and thus need detailed modeling steps, which are theoretically justified by the ontologies. We developed the two ontologies to specify knowledge content and the functional-knowledge modeling process.

3. Ontologies to Capture Functionality

Fig. 1 outlines our framework for functional modeling based on the ontologies. It has two levels, i.e., the behavioral and the functional ('a' axis). The extended device ontology [6] provides fundamental concepts such as "**device**" mainly for the behavioral level. (Terms in bold letters are defined in the ontology). The functional concept ontology [7] provides a vocabulary for describing the functional-level model and maps between behaviors and functions.

At the behavioral level, the model is objective without the designers' intentions. It consists of **devices**, **connections** between devices ('b' axis), **assembly** (or aggregation) relations of devices ('c' axis), and **behaviors** of entities. The "behavior" of a device is defined as the objective interpretation of its input-output relation considering it as a black box. Each device is connected to another through its input or output **ports**. A device plays a role as an **agent** (or actor) that changes the states of what is input (called **operand**, i.e., what is being processed by the device) such as fluid, energy, motion, force, and information. The input-output relation of the behavior is, more precisely, the difference between the states of the operand at the input port and that at the output port (called **IO-State)**. A device can be a mechanical element, a mechanical pair, a component, an assembly, or a system.

Fig. 2 shows definitions of such concepts in the extended device ontology in an ontology editor of an environment for building/using ontology named Hozo [26]. The ontology editor basically supports frame-based representation with slots. Concepts are represented as frames (denoted by nodes in Fig. 2) with slots (right-angled link) and the *is-a* relations among concepts (straight link with “is-a”). Concepts are categorized into the *wholeness* concepts composed of part concepts (shown in the left pane in the screen snapshot in Fig. 2) and the *relation* concepts between the concepts (the right pane). A wholeness concept has slots of *part* concepts (*part-of* relation denoted by right-angled link with “p/o”) and slots of attributes (“a/o”). A relation concept has slots of *participant* concepts (*participate-in* relation. denoted by “p/i”) and the attribute-slots. One of the features of Hozo is theoretical treatment of *role* concepts with slots. By role we mean here such a concept that an entity plays in a specific context and cannot be defined without mentioning external concepts [26], which is similar to the definitions in the literature [27,28]. For example, a man (class constraint for role) can play “husband role” (*role* concept) in a “marriage” relation (role context), who is called “husband” (*role holder*). It can be defined as that the “marriage” relation concept has two *participate-in* slots; i.e., the “husband” slot (“man” as a class constraint, “husband-role” as a role concept and “husband” as a role-holder) and the “wife” slot (“woman”, “wife-role” and “wife”, respectively). These roles can be defined also with part-of relation of the “married couple” which is a *wholeness concept* corresponding to the “marriage relation”. For details of Hozo, see [26].

In the definition of the extended device ontology in Fig. 2, the **device** concept is defined as a *role-holder* in **behavioral-relation** between two **physical-entities** (Note that we assume this concept is a sub-class of more generic concept in an upper ontology). One of them plays the “agent” role, which is called **device**. It operates the other entity (**operand** which is another *role-holder*) and changes its physical attributes. The

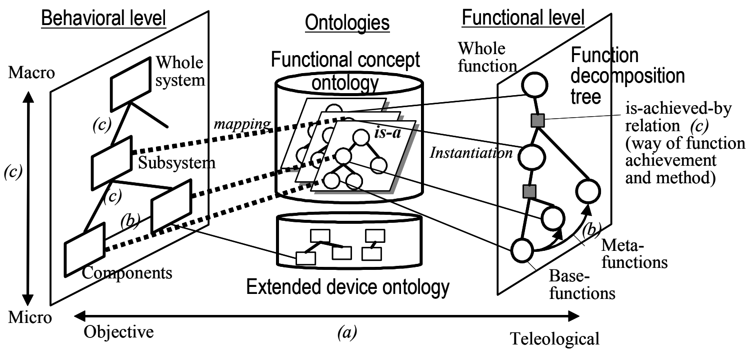


Fig. 1. Structure of a Functional Model

physical-entity as an operand has **IO-States**, which represents values of **physical-attribute**s at a port of a device. The pairs of IO-States at input ports of a device and those at output ports of the same device are defined as **behavior**. A physical-entity has a set of kinds (denoted by #) of physical-attributes (i.e., not instances of physical-attributes but pointers to the class) for description of qualitative relations between the physical-attributes.

We extended the conventional device-centered ontologies (e.g., in [9,11,23]) originating from systems dynamics theory by redefining the concepts of behavior, **conduit**, and **medium**. We categorized the meanings of behavior into four types [6]. The definition above (called B1 behaviour) is distinguished from the other three for capturing functionality. A **conduit** (e.g., a pipe and a shaft) is defined as a special device that transmits an operand without any change in an ideal situation. A **medium** (e.g., steam for heat energy) is something that holds an operand and enables it to flow between devices. In Fig. 2, **medium** is defined as a role-holder which carries another physical-entity(operand) in **carrying-relation**. The refined definition enables us to cope with mechanical domains that seemingly do not fit device ontology [29].

The functional level represents the "teleological" description of a system with the designer's intention. We define a "**function**" of a device as a conceptualization of the teleological interpretation of its "behavior" with the intended goal [7]. We have defined about 220 generic functions such as "to give energy" and "to split things" (called functional concepts) in the functional concept ontology. The definition is in terms of FTs (Functional Toppings), which represent information about the teleological interpretation of (mapping to) a behavior according to the designers' intentions.

The vertical axis denoted 'c' at the functional level in Fig. 1 represents aggregation (or decomposition) of functions, that is, a sequence of micro(sub)-functions achieves a macro(whole)-function, which we call the "is-achieved-by" (a kind of *part-of*) relation. It corresponds to function decomposition [5], whole-part relation [19] and "degree of complexity" [3]. In addition to such a description of "how to achieve the func-

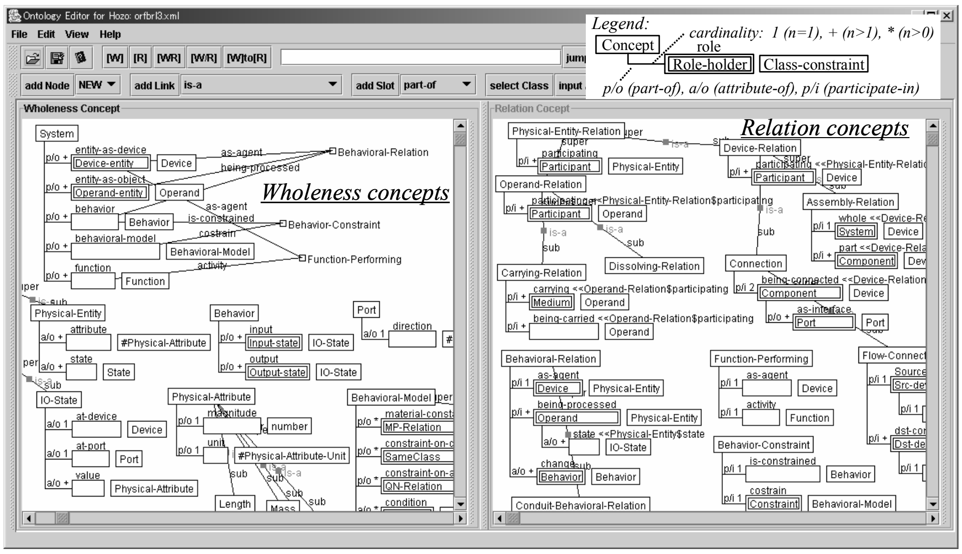


Fig. 2. Portion of the extended device ontology defined using Hozo

tion" (we call a *method*), the concept "*way of function achievement*" represents the conceptualization of background knowledge for function decomposition such as physical principles, which represents "why the sequence of micro-functions can achieve the macro-function". The conceptualization of the *way* concept helps us distinguish "how to achieve and why" (way) from "what is intended to be achieved" (function). For example, the example of "to weld" in Section 2 can be described as *fusion way* of the *joining function*. The fusion way has specific characteristics of the output that the operands are fused and they are hard to be separated. Although a functional concept "to join" loses some amount of information of "to weld", what is loses goes to the characteristics of the fusion way. As a total, functional concepts are successfully made very generic without any loss of information. In the fusion way, the joining function (a macro-function) can be achieved by three micro-functions; "make distance between operands zero", "melt parts of them" and "solidify them". The heating function in Section 2 is the sub-function of the melting function and can be achieved in the arc way. How to describe such function decomposition tree will be discussed in the following section using another example.

In Hozo, a functional decomposition tree is described as a model composed of instances of the functional classes defined in the ontologies. For example, functions in a functional decomposition tree are instances of the generic functional-concept classes and should satisfy necessary conditions of their definitions. The concept of "way of function achievement" is defined as a relation concept between functions. It governs the aggregation relations between functions.

4. Ontology-based Modeling Process

On the basis of these two ontologies as theoretical background, we have developed a modeling methodology that consists of types of functional knowledge, specifications for modeling processes (Fig. 3), and guidelines for descriptions (Table 1). Fig. 3 outlines a modeling process from a functional model of a concrete artifact to organized generic knowledge. Each node represents an activity by the knowledge authors at each step. An activity consists of some sub-activities in sub-steps (this task decomposition is denoted by lines with diamonds from left to right). Table 1 lists some of the guidelines for describing the function decomposition tree based on the ontologies. Here, we use a production machine called a wire-saw as an example, which is shown in Fig. 4. This is adapted from the deployment discussed in Section 5. It is designed to slice semiconductor ingots with friction by moving wires. We extended the rough steps reported in [30] and clarified the guidelines.

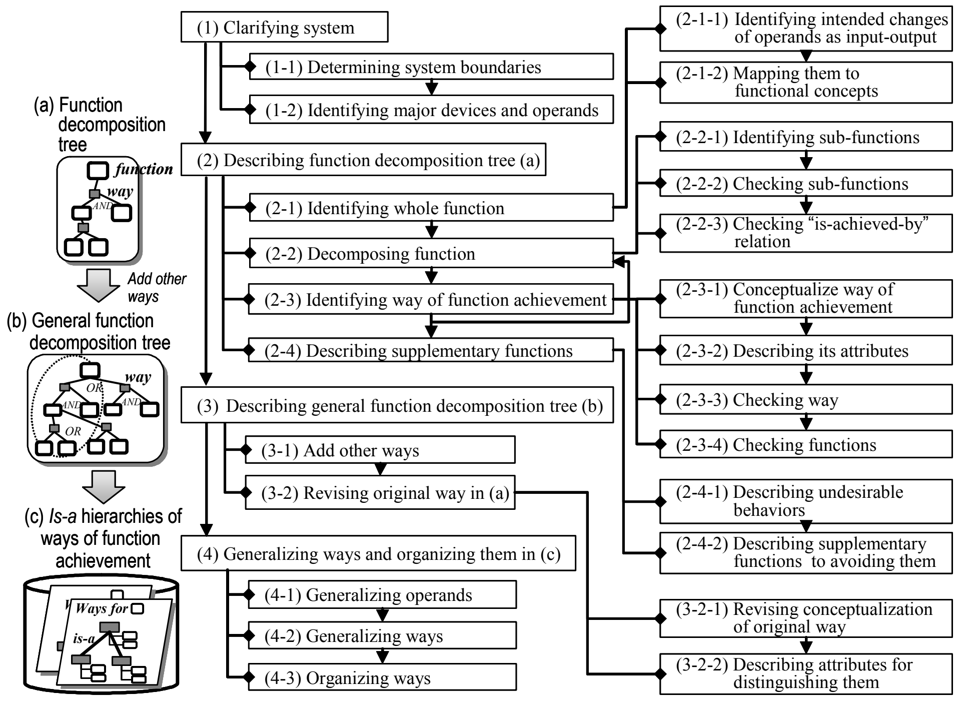
4.1 Clarifying System

The first step (#1 in Fig. 3) involves analyzing the system to be described and clarifying it. The first sub-step (#1-1) involves determining the boundaries for the model, i.e., criteria for judging whether a thing (a component etc.) will be modeled or not. If not, it will be treated as an external factor to the modeled system. The boundaries are spatial and temporal. The temporal boundary is important to distinguish the design process, manufacturing process, and product functioning process as shown in the guidelines F2 in Table 1.

The second sub-step (#1-2) is to identify physical things participating in the process (called participants) in the boundaries and then assign a role to each of them according to the extended device ontology discussed in Section 3 and guidelines F2, F3, and S3. Because decomposition has not yet been done at this point, the major (larger grain-sized) components (devices) are identified. In the wire-saw example in Fig. 4, the major components include the motor and the roller. The ingot is obviously an operand. However, the wire can be a problem in that it can be considered an *agent* (to exert force on ingots), an *operand* (to be moved by roller), or a *conduit* (to transmit tension). According to the semantic constraints in the extended device ontology, one possible consistent role-assignment is to decompose the wire into two parts, a working wire as an agent and a transmitting wire as both a medium and a conduit. The extension of the device ontology accepts the last situation.

4.2. Describing Function Decomposition Tree

The second step (#2) is to describe the *function decomposition tree* of the target system (denoted (a) in Fig. 3) at the functional level in Fig. 1. It consists of a macro-function, sub(micro)-functions, relations between sub-functions, and *ways of function achievement*. Figure 4 shows (a) the initial model and (b) the revised one for the function decomposition tree of the wire-saw.



(a) Function decomposition tree
function
way
AND
Add other ways
(b) General function decomposition tree
way
OR
AND
AND
OR
(c) Is-a hierarchies of ways of function achievement
Ways for
is-a
(1) Clarifying system
(1-1) Determining system boundaries
(1-2) Identifying major devices and operands
(2) Describing function decomposition tree (a)
(2-1) Identifying whole function
(2-2) Decomposing function
(2-3) Identifying way of function achievement
(2-4) Describing supplementary functions
(3) Describing general function decomposition tree (b)
(3-1) Add other ways
(3-2) Revising original way in (a)
(4) Generalizing ways and organizing them in (c)
(4-1) Generalizing operands
(4-2) Generalizing ways
(4-3) Organizing ways
(2-1-1) Identifying intended changes of operands as input-output
(2-1-2) Mapping them to functional concepts
(2-2-1) Identifying sub-functions
(2-2-2) Checking sub-functions
(2-2-3) Checking "is-achieved-by" relation
(2-3-1) Conceptualize way of function achievement
(2-3-2) Describing its attributes
(2-3-3) Checking way
(2-3-4) Checking functions
(2-4-1) Describing undesirable behaviors
(2-4-2) Describing supplementary functions to avoiding them
(3-2-1) Revising conceptualization of original way
(3-2-2) Describing attributes for distinguishing them

Fig. 3. Modeling Process of Functional-Design Knowledge

(1)Identifying Whole Function

The first sub-step (#2-1) is to "identify the whole function" of the system according to F1-F4. Here, "*the way of function achievement*" discussed in the previous section plays an important role. In the example, the whole function is not "to slice" but "to split", because "to slice" implies "how to split" and provides specific information about the thinness of the split part. The former information is regarded as the *way of function achievement*. The goal of slicing here can be considered to be "to split a part from the target operand (i.e. ingot)". It makes it possible to select other *ways* instead of "slicing" in the design. In reality, slicing with wire is not single *way of function achievement* but a composite as will be discussed later.

The latter information (i.e., thinness), on the other hand, is regarded as the quantitative degree of a function. Each *way of function achievement* has specific value of attributes like it. Then, such information can be used as conditions to select the *way* from all available *ways of function achievement.*

Table 1. Guidelines for function decomposition tree

F. About functions and behaviors

F1. A function represents "what to achieve" only and does not imply "how to achieve".
  F1-1. A device is a black-box. The inside is not shown at a level.
F2. A function represents (a teleological interpretation of) changes in physical things within the system boundary.
  F2-1. Do not describe the designer's activities.
  F2-2. Distinguish product's functions, manufacturing processes, and recycling activities.
  F2-3. Determine a system boundary with a pre- and post-process.
F3. Agent of functions should be a "device" in the physical world.
  F3-1. A human operator can be regarded as a "device".
  F3-2. Designers and manufacturer should be distinguished.
  F3-3. Sizes of devices decrease in function decomposition.
  F3-4. A device can be virtual and dynamic.
F4. Decompose functions which imply kinds of operands and/or degrees of results for functions.
  F4-1. Such implications are represented as attributes of *ways of function achievement*

S: About relations between sub-functions

S1. Identify states of operands that flow sub-functions.
S2. Time passes along this relation.
S3. Roles of things as operands should not be changed in a series of functions.

A: About "is-achieved-by" relation and way of function achievement

A1.The "is-achieve-by" relation represents aggregation
  A1-1. The total changes in sub-functions should correspond to changes in the whole function.
  A1-2. This relation does not imply a time interval.
  A1-3. This relation is not an "is-a" relation.
A2. A sub-function should explicitly contribute to a macro-function.
  A2-1. Explicate implicit sub-functions.
A3. The *way of function achievement* represents a single principle.
  A3-1. Decompose compound principles
  A3-2. Distinguish them from other ways at the principle level.
  A3-3. If possible, conceptualize neither tools nor operands but principles
  A3-4. A way should refer to a direct macro-function.
A4. Distinguish supplementary functions from essential functions.

(2) Decomposing Functions

The second sub-step (#2-2) involves decomposing the whole function (generally, a macro-function) into sub (micro)-functions that can achieve the macro-function. When one regards it as a design activity, it corresponds to function decomposition [5]. After the modeler has tried to identify the sub-functions (step #2-2-1), important steps are to check the relations among sub-functions (as step #2-2-2) according to the S1-S3 guidelines and to check the relations between sub-functions and the whole-functions (as step #2-2-3) according to the A1-A4 guidelines.

As we can see from Fig. 4(a), one might describe “to move table to wire” and “to move wire” as sub-functions. However, against A2, why these two sub-functions can perform the whole function is not clear. Moreover, against S2, it is unclear which operands flow between the two sub-functions. One reason is that there is a missing sub-function, “to exert vertical force to ingot and wire”. The original sub-function “to move table to wire” contributes to the vertical-force sub-function. The other missing sub-functions are found in the next sub-step.

(3) Describing Ways of Function Achievement

The third sub-step (#2-3) involves describing the *ways of function achievement*. The modelers identify a physical principle that can achieve the whole(macro-) function and conceptualize it (#2-3-1). Then, the attributes of the *way* are described (#2-3-2). Next, such descriptions are checked according to A3 (#2-3-3). As a result of identifying the *way*, the functions are sometimes changed (#2-3-4). To further decompose sub-functions, steps #2-2 and #2-3 are done recursively.

In the wire-saw example, the *wire-saw way* does not involve a single *way* but a composite of three *ways*, i.e., the removal way of splitting, the physical force way of losing combinatorial force of a part (kerf loss, i.e., the part lost by cutting), and the linear friction way of exerting force. Splitting is achieved in two sub-functions; losing the combinatorial force of the kerf-loss and moving it away. This *way* to achieve the function is conceptualized as the removal way based on separating the kerf loss part.

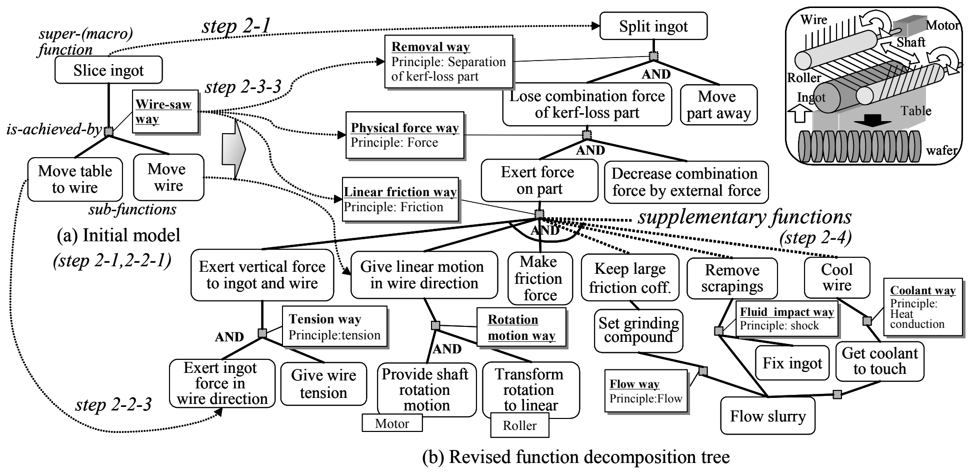


Fig. 4. A function decomposition tree of a wire-saw for slicing ingots (portion)

(4) Describing Supplementary Functions
The last sub-step (#2-4) involves adding *supplementary functions* that are not essential but provide additional effects to improve efficiency and/or to prevent faults. In other words, we recommend describing essential sub-functions only until this step, because this clarifies the principles for achievement. In this paper, we have omitted this aspect but it is reported in another [31].

4.3. Describing General Function Decomposition tree

A *general function decomposition tree* (b) includes possible alternatives to achieving functions in an OR relationship. In the first sub-step (#3-1), the function decomposition tree described in the previous step (#2) is expanded by adding other *ways of function achievement* for each function decomposition. Then, the *way of function achievement* in the original function decomposition tree are revised by comparing with principles for the other *ways* (#3-2). This step can be omitted.

4.4. Generalizing Ways of Function Achievement

A concrete *way* in a (general) function decomposition tree can be generalized into a generic way in steps #4-1 and #4-2. Generic *ways* are called functional way knowledge and they consist of a macro-function, a set of sub (micro)-functions, temporal and causal constraints among sub-functions, principles of achievement, conditions for use of the way (e.g., the specific class of operands (e.g., solid objects) which can be changed in the way), and quantitative characteristics of the *way* (e.g., accuracy, cost, time, amount of change (e.g., limitation of thinness for splitting)). Although this includes a description of the method to achieve functions, we called it the "*way*", focusing on the fact that it includes a description of the principles of achievement.

Then, *ways* to achieve the same function are organized in *is-a* relations according to their principles (called an *is-a hierarchy of ways of function achievement* (c) in Fig. 3 and Fig. 5) in step #4-3. We distinguish the organization as an *is-a* hierarchy from the other derivative organizations depending on the viewpoint. Such ad-hoc trees can be reorganized by a functional-way server according to the given viewpoint [32].

Figure 5 details *is-a* hierarchies of *ways* to achieve split functions and others. They have been generalized from the specific *ways* used in the wire-saw example in Fig. 4 and from other cutting machines such as water-jet cutting and electrolysis cutting. Conventional organization of *ways* of cutting in a textbook of the field relies on "what is used for" against guideline A3-3. Figure 5 shows explicit differences between the wire-saw and other cutting devices. The wire-saw uses the three *ways* marked with asterisks. Moreover, *ways* of exerting force can also be used in other appliances, e.g., washing machines. In a screw-type washing machine, for example, dirt is separated from cloth by random frictional force caused by the rotating screw. This kind of knowledge is general and can be applied to different domains.

4.5. Types of Knowledge

Note that these types of trees concerning functions (Fig. 3) are different. Function decomposition tree (a) represents *is-achieved-by* (a kind of *part-of*) relations among functions. The *is-a* hierarchies of ways (c) represent an abstraction of the key information about how to achieve the function, while the *is-a* hierarchies in the functional concept ontology represent abstractions of functions themselves, i.e., the goals that are achieved. Moreover, there are a huge numbers of ways for a function in nature, while the numbers of functional concepts are small.

The modeling process discussed in this section is used to describe functional knowledge from the bottom-up from scratch. When the general functional way knowledge is available, the modeler can use this to describe the function decomposition tree and/or add a new way of function achievement to an existing general function decomposition tree or an existing *is-a* hierarchy. Moreover, the steps; #2-1 and #2-2 can be done in reverse, i.e., from micro-functions to macro-functions from the bottom up. The functions of components can be aggregated into macro-functions. In reality, both directions are mixed in the modeling process.

5. Deployment

The ontology-based modeling methodology discussed thus far has been deployed since May, 2001 at the plant and production systems engineering division of Sumitomo Electric Industries, Ltd. (hereinafter referred to as SEI) [8]. A knowledge management software named SOFAST has been developed based on part of the methodology and then deployed since December, 2002. Currently about 50 people in three factories use SOFAST in their daily tasks. The targets are manufacturing equipment mainly used in semiconductor manufacturing processes including the wire-saw shown in Figure 4, a wafer polisher, an optical fiber connector adjusting machine, and inspection machines. SOFAST has been used by 13 other companies since April, 2003 some of which use it in actual work. The followings summarize some of usages and

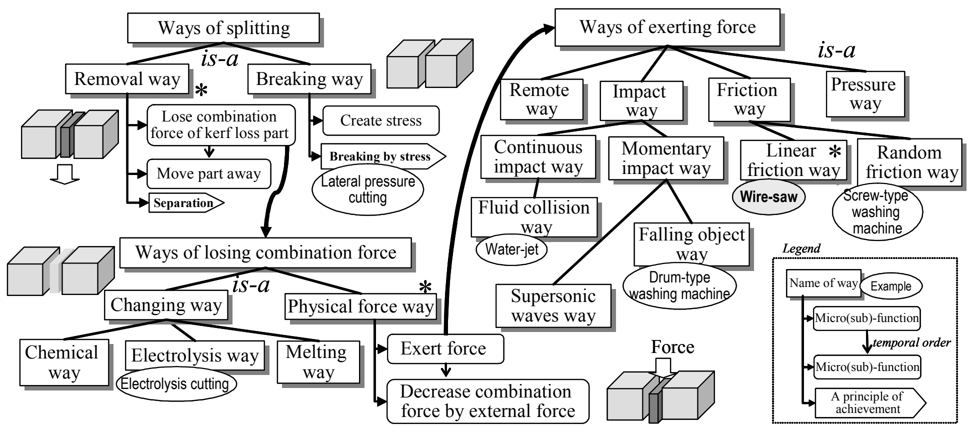


Fig. 5. Example of organizing generic ways of function achievement in *is-a* hierarchies

effects in the deployment.

SOFAST is designed to support the description of functional knowledge and sharing the knowledge in an intra-network. It consists of client software and knowledge repositories. Using the client software, a user can describe function decomposition trees through a graphical user-interface and store them in the repository. Then, all users can search *ways of function achievement* in the repository to achieve the function of interest by specifying a goal function.

One of use of the function decomposition tree is to clarify functional knowledge, which is implicitly possessed by each engineer, and share it with other engineers. The experiential evaluation by Sumitomo's engineers was unanimously positive. Writing a function decomposition tree according to the methodology gives designers the chance to reflect on good stimuli, which leads them to an in-depth understanding of the equipment. This is because such a function decomposition tree shows the designer's intentions on how to achieve the goal function and justify design decisions, which are not included in the structural or behavioral models.

Such a deep understanding contributes to redesigning and solving problems with the equipment. For example, an engineer was not able to reduce the time a machine requires to polish semiconductor wafers after four months of investigation by adjusting the known working parameters. He consequently described its function decomposition tree, by referring to that of the wire-saw in Fig. 4. Although these two devices have the different main functions, he found the shared function "to maintain a large friction coefficient" and its sub-function "to place diamond powder between wafers and the table". As a result, he became aware of an implicit function and its parameters for placing more diamond powder to obtain a high friction coefficient. Eventually, he reduced the necessary time to 76%, which was better than the initial goal. This improvement was achieved within three weeks.

The general function decomposition tree can be used to compare design candidates by explicating different ways to achieve functions. It contributes to patent analysis and patent applications. In communications between engineers and patent attorneys in applying for a new patent, it is difficult to determine the product's originality and to make appropriate claims. When the general function decomposition tree has been adopted as regular document format of documents for a patent application, the period was reduced to just one week from three or four weeks. Moreover, the patent claims were increased and doubled in some cases, since the attorneys found extra differences with other patents by checking at each level of function decomposition. The same benefit was found by another company in the users' group.

Generic knowledge about ways of function achievement help designers search ways to achieve a function and/or alternatives in an existing product. In deployment, a novice engineer developed an inspection machine in three days by systematically consulting generic ways of shedding light in the knowledge repository of SOFAST. Such development usually requires experts two weeks.

The success factors for deployment can be summarized as (1) clear discrimination between function (goal) and way (how to achieve the goal) and (2) clear discrimination among the *is-a* relation of functions, that of ways, and the *is-achieved-by* (a kind of *part-of*) relations of functions. The modeling steps and guidelines based on these discriminations provide hints to users to interpret how a device works consistently.

6. Related Work and Discussion

The target knowledge of this research is functionality of physical artifacts. It is "domain" knowledge of design problem-solving or diagnosis. It is different from "task" knowledge of designing or diagnosing, which is activity of human or automated problem-solvers. In the task ontology research, generic tasks and generic problem-solving methods (PSMs) are proposed (e.g., [33]). If one ignores the difference between domain and task, the generic tasks and the generic methods are similar to our generic functions and generic ways of function achievement, respectively. We focus on structuring knowledge about how to achieve functions (activities in domain world). We conceptualize the principle behind the sequence of activities (called *method* in both researches) as the *way of function achievement*. It helps us organize them in *is-a* hierarchies, though PSMs for a specific task are usually not organized well. Moreover, we distinguish function at the teleological level from behaviors at the objective level.

Behavior of artifacts in our work is a kind of "process" by which we intuitively mean a sequence of state changes over time. We concentrate on *physical* process which represents temporal changes of physical quantities as we discuss in the following paragraphs. On generic "process", extensive research has been done. The process specification language (PSL) [34] defines "activity" as a basic concept and temporal relationships between them. Although it has the theory on sub-activities, it includes neither the concept of *way* nor generic activities. *Formal ontologies* for processes have been investigated (e.g., [35]). The MIT process handbook treats business activities [36]. It includes taxonomy of basic business activities. Some activities such as "buy in a store" in specialization of activities, however, imply "how to achieve" like "welding" in Section 2. It obstructs organizing "how to achieve" (the way of function achievement in our methodology) separately from specialization of activity itself.

The *is-achieved-by* relation in our work is a kind of the *parthood* relation between functions. Parthood has been extensively investigated in the *formal ontology* research. For example, DOLCE ontology includes formal specifications of *parthood* (and other fundamental concepts) [37].

A great deal of work on domain ontologies in the engineering domain has been done [4,10,11,23,24]. We concentrated on "ontology as meta-knowledge", which provides knowledge authors with constraints and guidelines on capturing the target world, though pioneering work [10] in ontological research and extensive work in semantic web research aimed at "ontology for agent communication" that contributes interpretabilities among agents. Borst et al. proposed PhysSys ontology as a sophisticated lattice for ontologies for the engineering domain [11]. Although their's did not include ontology for functionality, we focus on this. Ontological consideration on functionality can be found in the literature [4,23,24]. Chandrasekaran and Josephson identified a device-centric function and an environment-centric function [4]. Our definition of function is device-centric.

Our main point was to clarify several relationships related to functionality, i.e.,

(A) The teleological interpretation of behavior as a function (axis (a) in Fig. 1),
(B) The *is-a* hierarchy of functions,
(C) The *is-achieved-by* (part-of) relations among functions (axis (c) in Fig. 1, Fig. 3(a), Fig. 3(b), and Fig. 4),
(D) The *is-a* hierarchy of *ways of function achievement* (Fig. 3(c) and Fig 5)

The relationship between function and behavior in (A) is similar to “means and ends” [19], the F-B relationship [20], and “aims-means” (including design requirements as well) [3]. On the other hand, papers such as [21] define that “behavior” is how to achieve a function (C) where distinction between behavior and function is relative. De Kleer defines function as a causal pattern between variables [15]. We identified operational primitives as FTs to represent intentions and then gave them operational definitions. Many “verb+noun”-style functional representations (as in Value Engineering [22]) lack such operationality. The recent efforts toward a standard taxonomy for engineering functions by the NIST Design Repository Project [14] are well established; however, they lack an operational relationship with behaviors and ontological specifications.

Concerning the *is-a* hierarchy of functions (B), the few (4-16) generally-valid functions [5] are too abstract to describe details of designer’s intentions. The hierarchy of “degree of abstraction” [3] for functions represents the specialization of functions with additional conditions. These conditions, however, sometimes include a specific *way of function achievement* such as “transportation by sea” [3] in the same manner as “welding” discussed in Section 2. We separated their conditions for specialization into specific attributes for the *way of function achievement*. Our functional concept ontology includes the functions proposed by Keuneke [17] and similar functions in flow-based functional modeling [18,19].

The ways of function achievement in (C) and (D) is similar to the “means” [12,13]. However, they treated a product-specific model [12] or generic knowledge without explicit organization [13]. We investigated how to organize conceptualized generic ways of achieving functions in (D). Similar ideas on generic patterns to achieve functions are discussed in the literature [16,20,21]. As well as functional decomposition, the design prototypes [16] include structural decomposition and the function prototype [20] has physical feathers. Our description of *ways* tries to maximize generality by pointing out partial (and abstract) information on structure and behavior. Generic patterns or so-called design catalogs (e.g., in [5]) mainly concentrate on mechanical pairs. Generic teleological mechanisms (GTM) are modified in design based on analogy [21]. In our approach based on a limited set of functional concepts, the ways of function achievement are organized in *is-a* hierarchies (D). Designers can explore them on several abstract levels explicitly.

The TRIZ (TIPS) theory provides some patterns (or strategies) for inventions based on contradiction between two physical quantities [38]. We did not concentrate on design strategies but on the modeling schema. The TRIZ theory also pays attention to physical principles (effects), although we established a clear relationship between physical principles and functional structures.

Limitations with Ontologies and Application Domain
We cannot claim the completeness of concepts in our functional concept ontology. We applied the ontologies to modeling power plants, chemical plants and appliances as well as manufacturing machines. The models include changes in thermal energy, flow rate, ingredients of fluids, and force and motion of objects [7]. The current functional concept ontology can describe simple mechanical products, although it does not cover static force balancing and complex mechanical phenomena based on the shapes of objects.

7. Summary

An ontology-based knowledge modeling methodology was reported. It is domain-specific but is basically applicable to a great deal of knowledge about artifacts from the important viewpoint of functionality. It has been deployed successfully in industry. This paper discussed the detailed modeling process from specific models to generic knowledge in *is-a* hierarchies. The modeling steps and the guidelines based on ontologies help knowledge authors describe sharable knowledge clearly.

Our ontologies are operationally defined in an ontology editor as shown in Section 3. Using the editor, when a knowledge author describes a model or generic knowledge, the editor can check models against constraints defined in the ontologies (see [32] for detail). However, the current SOFAST software does not support such ontological constraints. Such functionality is being planned.

Acknowledgements: The authors would like to thank Tomonobu Takahashi, Kouji Kozaki, Yusuke Koji and Mariko Yoshikawa for their contributions. Special thanks go to Dr. Masayoshi Fuse, Mr. Masakazu Kashiwase and Mr. Shuji Shinoki of Sumitomo Electric Industries Ltd., for their cooperation in deploying our methodology in production systems.

References

1. Umeda, Y., Tomiyama, T.: Functional Reasoning in Design. IEEE Expert (1997) 42-48
2. Chittaro, L., Kumar, A.N.: Reasoning about Function and its Applications to Engineering. Artificial Intelligence in Engineering, 12 (1998) 331-336
3. Hubka, V., Eder, W.E.: Theory of Technical Systems. Springer-Verlag; Berlin (1998)
4. Chandrasekaran, B., Josephson, J.R.: Function in Device Representation, Engineering with Computers 16(3/4) (2000) 162-177
5. Pahl, G., Beitz, W.: Engineering design - a systematic approach. The Design Council (1988)
6. Kitamura, Y., Mizoguchi, R.: Ontology-based systematization of functional knowledge. Journal of Engineering Design, to appear (2004)
7. Kitamura, Y., Sano, T., Namba, K., Mizoguchi, R.: A Functional Concept Ontology and its Application to Automatic Identification of Functional Structures. Advanced Engineering Informatics, 16(2) (2002) 145-163
8. Kitamura, Y., Kashiwase, M., Fuse, M., Mizoguchi R.: Deployment of an Ontological Framework of Functional Design Knowledge. Advanced Engineering Informatics, to appear (2004)
9. De Kleer, J., Brown, J.S.: A Qualitative Physics based on Confluences. Artificial Intelligence, 24 (1984) 7-283
10. Cutkosky, M.R., et al.: PACT: An Experiment in Integrating Concurrent Engineering Systems. Computer (1993) 28-37
11. Borst, P., Akkermans, H., Top, J.: Engineering Ontologies. Int'l Journal of Human-Computer Studies, 46 (2/3) (1997) 365-406
12. Malmqvist, J.: Improved Function-Means Trees by Inclusion of Design History Information. Journal of Engineering Design, 8 (2) (1997) 107-117
13. Bracewell, R.H., Wallace, K.M.: Designing a Representation to Support Function-Means based Synthesis of Mechanical Design Solutions. In Proc. of the International Conference on Engineering Design (ICED) 01 (2001)

14. Hirtz, J., Stone, R.B., McAdams, D.A., Szykman, S., Wood, K.L.: A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts. Research in Engineering Design, 13 (2002) 65-82
15. De Kleer, J.: How Circuits Work. Artificial Intelligence, 24 (1984) 205-280
16. Gero, J.S.: Design Prototypes: A Knowledge Representation Schema for Design. AI Magazine, 11(4) (1990) 26-36
17. Keuneke, A.M.: A Device Representation: the Significance of Functional Knowledge. IEEE Expert, 24 (1991) 22-25
18. Chittaro, L., Guida, G., Tasso, C., Toppano, E.: Functional and Teleological Knowledge in the Multi-Modeling Approach for Reasoning about Physical Systems: A Case Study in Diagnosis. IEEE Transactions on Systems, Man, and Cybernetics, 23(6) (1993) 1718-1751
19. Lind, M.: Modeling Goals and Functions of Complex Industrial Plants. Applied artificial intelligence, 8 (1994) 259-283
20. Umeda, Y, Ishii, M., Yoshioka, M., Shimomura, Y., Tomiyama, T.: Supporting conceptual design based on the function-behavior-state modeler. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10 (1996) 275-288
21. Bhatta, S.R., Goel, A.K.: A Functional Theory of Design Patterns. In Proc. of IJCAI-97 (1997) 294-300
22. Miles, L.D.: Techniques of value analysis and engineering. McGraw-Hill (1961)
23. Kumar, A.N., Upadhyaya, S.J.: Component-Ontological Representation of Function for Reasoning about Devices. Artificial Intelligence in Engineering 12 (1998) 399-415
24. Salustri, F.A.: Ontological Commitments in Knowledge-based Design Software: A Progress Report. In Proc. of the 3rd IFIP WG 5.2 Workshop on Knowledge Intensive CAD (1998) 31-51.
25. Gruber, T.: A Translation approach to portable ontologies. Knowledge Acquisition, 5(2) (1993) 199-220
26. Kozaki, K., Kitamura, Y., Ikeda, M., and Mizoguchi R.: Hozo: An Environment for Building/Using Ontologies based on a Fundamental Consideration of “Role” and “Relationship”, Proc. of EKAW2002 (2002) 213-218
27. Sowa, J. F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole (2000)
28. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social Roles and their Descriptions, In Proc. of KR 2004 (2004)
29. Mortensen, N. H.: Function Concepts for Machine Parts - Contribution to a Part Design Theory, Proc. of ICED 99, 2 (1999) 841-846
30. Kitamura, Y., Mizoguchi, R.: Organizing Knowledge about Functional Decomposition. In Proc. of ICED 03, 2003
31. Koji, Y., Kitamura, Y., Mizoguchi, R.: Towards Modeling Design Rationale of Supplementary Functions in Conceptual Design. In Proc. of TMCE 2004 (2004) 117-130
32. Kitamura, Y., Mizoguchi, R.: Ontology-based description of functional design knowledge and its use in a functional way server. Expert Systems with Application 24 (2003)153-166.
33. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W. and Wielinga, B.: Knowledge Engineering and Management - The Common-KADS Methodology, The MIT Press, Cambridge, MA (2000)
34. ISO TC184/SC4/JWG8, Process Specification Language, Part 1, 11 and 12, http://www.tc184-sc4.org/SC4_Open/SC4_Work_Products_Documents/PSL_(18629)/ (2003)
35. Menzel, C., Gruninger, M., A Formal Foundation for Process Modeling, Formal Ontology in Information Systems; Collected Papers from the Second Int’l Conf. (2001) 256-269
36. Herman, G. A., Malone, T. W.: What is in the process handbook?, Organizing Business Knowledge: The MIT Process Handbook, MIT Press (2003) 221-258
37. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: The WonderWeb Library of Foundational Ontologies, WonderWeb Deliverable D17 (2002)
38. Sushkov, V.V., Mars, N.J.I., Wognum, P.M.: Introduction to TIPS: a Theory for Creative Design. Artificial Intelligence in Engineering 9 (1995) 177-189