Meta-Functions of Artifacts

Yoshinobu Kitamura and Riichiro Mizoguchi

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

Abstract

This paper proposes a new category of functions of artifacts called meta-function. Although conventional research on functional representation defines function as some kinds of abstraction of changes in objects associated with components, a meta-function of a component represents a role played by a function of the component for another function of another component without mention of such changes specific to the target system. The meta-functions explain the types of collaboration and dependency among functions of components as part of the design rationale of the target artifacts. This paper defines nine types of the meta-functions such as ToEnable and ToDrive, which form part of an ontology of functional concepts. The utility of the ontology in explanation and redesign is also discussed.

Introduction

Understanding of functionality is important for human understanding of artifacts, because functionality represents a part of the design rationale (Chandrasekaran et al., 1993; Lee 1997). A lot of research has been carried out on functional representation of artifacts. In literature, function of an entity (a component or a system) is defined as its intended behavior (Umeda et al., 1990; Lind, 1994), interpretation of its behavior under a goal (Sasajima et al., 1995), a kind of hierarchical abstraction (Sembugamoorthy and Chandrasekaran, 1986; Vescovi et al., 1993), or effects to the environment of the entity (Chandrasekaran and Josephson, 1996). Anyway, a function of an entity represents a kind of abstraction of changes in objects associated with the entity. We call them base-functions of components.

In this paper, we focus on collaboration among the base-functions of components in a target system. The components work collaboratively in order to make the whole system work. Each of them depends on each other, and has a role for each other. For example, vaporization (a base-function) of a boiler in a steam power plant can be said to **enable** the rotation of a turbine shaft. On the other hand, another function "to give heat" of the same boiler is said to **drive** the rotation. Roles such as "to enable" and "to drive" represent not only causal relations among parameters in these components at the behavioral level (such as a CPD in CFRL (Vescovi et al., 1993)) but also types of collaboration among the components at the functional level.

We call such a collaborative role a **meta-function**, which represents a role played by a base-function of a component for another base-function of another component in order to make the whole system work collaboratively. In these terms the first example mentioned above can be paraphrased as that the "to vaporize" base-function of the boiler is said to have a ToEnable meta-function for the "to rotate" base-function of the turbine. The definitions of such meta-functions in terms of both the behavioral causal relations and intended roles in collaboration among the base-functions are required.

Keuneke proposes four types of functions; ToMake, ToMaintain, ToPrevent, and ToControl (Keuneke, 1991). While her definitions are also concerned with the state in the component achieving the function, the meta-functions aim to explicate the objectives of the desired state or undesirable state, that is, which function the desired state contributes to, or which function the undesired state causes malfunctioning of. We distinguish between the first two and the rest and classify the former as types of a base-function and the latter as types of a meta-function.

This paper proposes nine types of meta-function, aiming at capturing collaboration among function. We have extended a representation language of behavior and functions of components named FBRL (Sasajima et al., 1995) in order to cope with meta-functions.

Such types of meta-functions form part of an ontology of functional concepts (Kitamura and Mizoguchi, 1998), which provides a rich and comprehensive vocabulary for functional representation. An ontology is defined as an explicit specification of conceptualization (Gruber 1993) and theory of a primitive vocabulary for knowledge-based systems (Mizoguchi and Ikeda, 1997). The functional concepts in our ontology are categorized into four spaces including the meta-function space, which are organized in is-a or part-of hierarchy. They are clearly defined by conditions of behavior and the additional information for teleological interpretation.

Explicit conceptualization of domain knowledge as ontologies helps several kinds of problem solving (Abu-Hanna and Jansweijer, 1994; Borst et al., 1997). We will mention utility of our ontology in explanation and redesign. Meta-functions can be used for explaining design rationale of organization of the target system without mention of changes of entities specific to the target system. They enable us to capture the dependency among base-functions in a target system at an abstraction level, which enables a redesign system to propose improvements of an existing artifact with consideration of such dependency in the original design as a way of the dependency management in the design task (Lee, 1997).
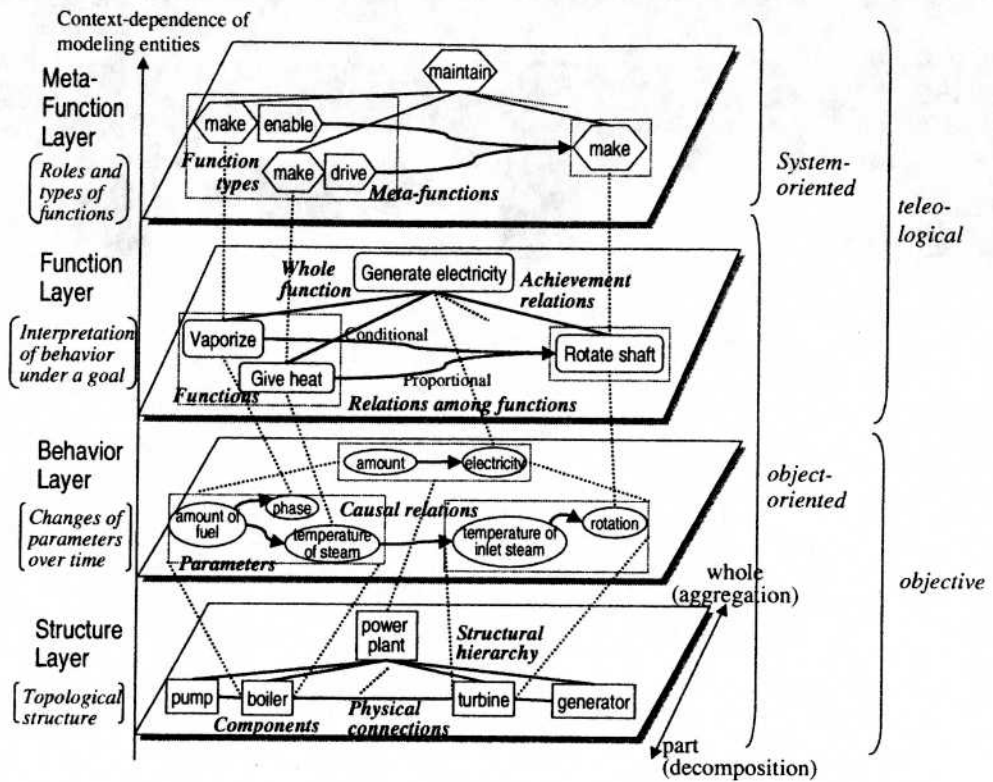
Figure 1: The four layers of an example model

Firstly, we describe the layers of our functional model and introduce the meta-function layer. Next, the ontology of functional concepts is overviewed. On the basis of such preparation, the types of meta-functions are defined. Next, the automatic identification of meta-functions is discussed. Then, the utility of the ontology and meta-functions is mentioned. The contribution of this work by comparison with the related work and the limitations of the ontology are also discussed.

Structure, Behavior, Base-Function and Meta-Function

This section overviews the general structure of our modeling scheme using a part of the model of a power plant shown in Figure 1 as an example. There are three axes. The vertical one has four layers. The layers of structure, behavior and (base-)function are similar to those proposed in literature such as (Lind, 1994). The meta-function layer is introduced here. The horizontal axis represents relations among entities of the same grain size. The last axis to the depth represents the grain size of the entities. An entity is a part of deeper one. In each layer, there is a whole-part hierarchy.

Structure layer and Behavior layer. The structure layer describes topological structure of the artifact. We adopt the device-oriented ontology (de Kleer 1984). It describes the existence of components, physical connections among them, and a structural hierarchy. The behavior layer represents changes of parameter values of entities over time. The horizontal relations represent causal relations among parameters. The whole-part relations represent hierarchical abstractions of behavior.

Base-Function layer. A base-function of a component is defined as a result of interpretation of a behavior of the component under an intended goal (Sasajima *et al.*, 1995). The functions of the boiler includes "to vaporize water" and "to give heat to water" as shown in Figure 1. The horizontal relations among functions are causal or structural. The causal-type relations are categorized into some sub-types such as proportional and conditional according to causal relations among parameters that functions focus on. The former represents that a parameter value has proportional relation with another parameter. The latter represents the case where the condition is discrete such as the condition that the phase of the steam has to be gas for a turbine. On the other hand, the structural-type relations are categorized into subtypes such as series, parallel, sequential, simultaneous and feedback. The whole-part relations represent that the whole functions are achieved by groups of sub-functions (called achievement relations).

Meta-Function layer. The meta-function layer describes the roles of each function for another function (called meta-functions) and the types of functions (called function types). For example, the "to vaporize" function of the boiler is said to be ToMake-type and to perform a meta-function ToEnable for the "to rotate" function of the turbine according to the definition discussed later. Such a meta-function refers not changes of objects of these components but functions of the components, while other three layers are concerned with existence or changes of objects.
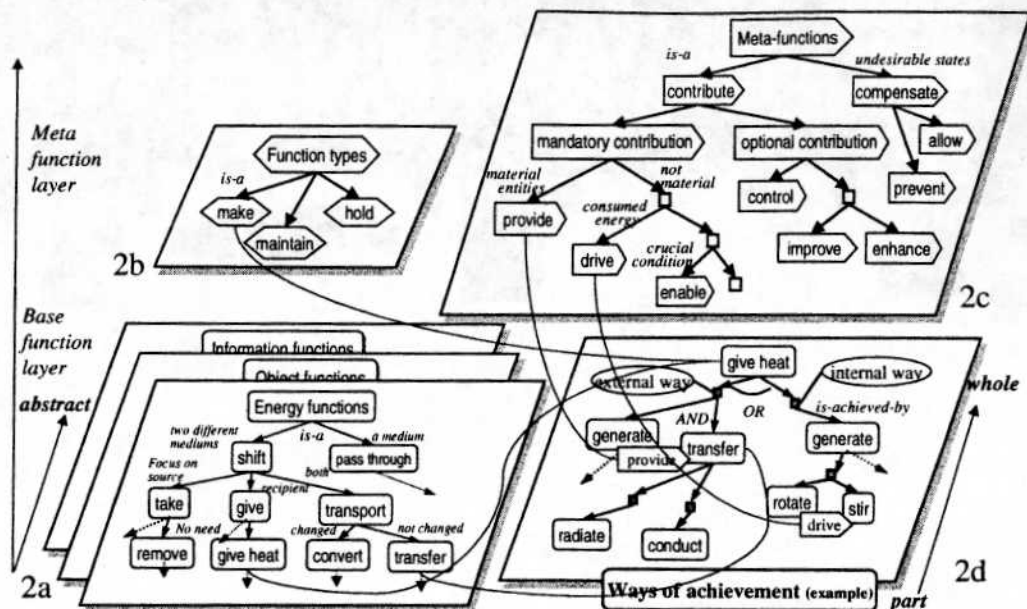
Figure 2: The four spaces of the ontology of functional concepts (part)

Ontology of functional concepts

The ontology of the functional concepts is designed to provide a rich and comprehensive vocabulary for both human designers and design supporting systems. It consists of the four spaces as shown in Figure 2. In this section, we explain three spaces other than the space of the meta-function discussed in the next section.

Base-functions

A base-function of a component can be represented by a transitive verb of which grammatical subject is the component and of which grammatical objects are the incoming or outgoing entities of the component. Although a function depends on the context, the description itself should be local. A behavior can be interpreted from object-related aspect (called *object functions*), energy-related aspect (called *energy functions*) or information-related aspect (called *information functions*). For example, the object function and the energy function of a turbine are "to rotate the shaft" and "to generate kinetic energy", respectively.

Figure 2a shows the energy base-functions organized in an is-a hierarchy with clues of classification. A base function is defined by conditions of behavior and the information for its interpretation called Functional Toppings (FTs) of the functional modeling language FBRL (abbreviation of a Function and Behavior Representation Language) (Sasajima *et al.*, 1995). There are three types of the functional toppings; (1)O-Focus representing focus on attributes of objects, (2)P-Focus representing focus on ports (interaction to neighboring components), and (3)Necessity of objects.

For example, a base-function "to take energy" is defined as "an energy flow between two mediums" (a behavioral condition), and "focus on the source medium of the transfer" (functional toppings). Moreover, the

definition of "to remove" is that of "to take" plus "the heat is unnecessary". Thus, "to take" is a general (super) concept of "to remove" as shown in Figure 2a.

The values of O-Focus and P-Focus represent intentional focus of the function. Such a parameter that is a kind of O-Focus and is an attribute of the entity in the focused port indicated by P-Focus is called as the focused parameter. The entity (object or energy) having the focused parameter is called as the focused entity.

The object functions are categorized into the following two categories according to the kinds of the focused parameter. The one is called as the *amount function* which changes the amount of the target objects or changes the kind of the objects. The other is called as the *attribute function* which changes the other attributes of the objects. These categories are used in the definitions of the meta-functions.

Note that definition of base function using FTs is highly independent of its *realization* by which we mean that details of behavior and internal structure of the component. For example, P-Focus specifies not concrete location but abstract interaction with the neighboring components. The realization is represented by the ways of achievement shown in Figure 2d.

Function Types

The function types represent the types of goal achieved by the function (Keuneke, 1991). We have redefined the function type as "ToMake", "ToMaintain", and "ToHold". For example, consider two components, an air-conditioner (as a heating device) and a heater, having the same function "to give heat". The former keeps the goal temperature of a room. The latter does not. These are said to be "ToMaintain" and "ToMake", respectively.

Ways of Achievement

A base-function f_u can be achieved by the different groups of sub-functions. We call a group of sub-

	Object/Energy (focused entity of f_a)	Mandatory (f_a is mandatory for f_t)	Type of relation (between f_a and f_t)	Material (the focused entity of f_t is made of the focused entity of f_a)	Consumption (the focused entity of f_a is consumed by f_t)
Provide	*	✓	*	✓	*
Drive	Energy	✓	Proportional	✗	✓
Enable	*	✓	Conditional	✗	✗
Improve	*	✗	Proportional	*	*
Enhance	Energy	✗	Proportional	*	*

Legend: ✓ must be, ✗ must not, * don't care, f_a : the agent function, f_t: the target function

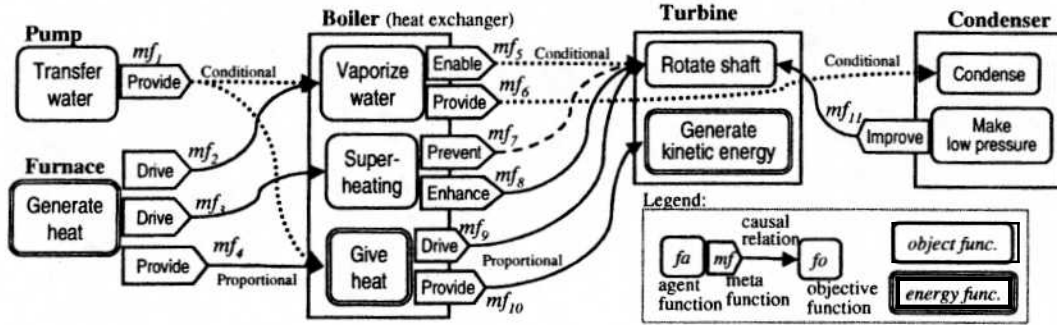Table 1: The necessary conditions for meta-functions (part)

Figure 3: Meta-functions in a power plant (part)

functions $\{f_1, \dots f_n\}$ constrained by the relations among them (such as meta-functions) *a functional method* of an achievement of f_u. On the other hand, we call the basis of the method *a functional way*. The way is the result of conceptualization of the physical law, the intended phenomena, the feature of physical structure, or components used.

Figure 2d shows some ways of achievement of "to give heat to an object", which are described in terms of concepts in other three spaces. There are two ways, that is, the external and internal heat source ways. According to the external heat-source way, it is decomposed into two sub-functions, that is, "to generate heat" and "to transfer heat". The former should perform a ToProvide meta-function for the latter. In the figure, the latter function can be decomposed according to "radiation way" or "conduction way".

Note that Figure 2d shows is-achieved-by (whole-part) relations among the functional concepts in OR relationship, while Figure 2a shows is-a relations as the definitions of them, which are independent of "how to realize them".

Meta-Functions

The meta-functions (denoted by *mf*) represent a role of a base function called an **agent function** (f_a) for another base function called a **target function** (f_t). We have defined the nine types of meta-functions as shown in Figure 2c (an is-a hierarchy) and Table 1 (the necessary conditions). Figure 3 shows examples of the meta-functions $\{mf_1 \dots mf_{11}\}$ in a simple model of a power plant (part). Note that the furnace which is a sub-component of a boiler is separated from the boiler as a heat exchanger for explanation.

The conditions for the meta-functions refer the definitions of the agent and target base-functions and the relations among them. The focused entity (object or energy), the focused parameter, and the categories of the base-functions such as the amount function and the attribute function defined in the previous section play a crucial role in the definitions.

We begin with the condition where there is a causal relation from the focused parameter of f_a to that of f_t. If the goal of f_t is not satisfied when f_a is not achieved, the f_a is said to have a mandatory contribution for the f_t (see Figure 2c). Although we can intuitively say that the f_a has a ToEnable meta-function for the f_t, the authors define a narrower meaning of ToEnable by excepting the cases of ToProvide and ToDrive as follows.

ToProvide (Provider role)

When a function f_a generates (or transfers) the *materials* which another function f_t intentionally processes, the function f_a is said to perform a meta-function "to provide material" for the function f_t. The material of f_t can be basically defined as input objects (or energy) which will be a part of the output objects on which the function f_t focuses. If f_t is an amount function (see the previous section for the definition), the output objects are mixture of the source materials or synthesized from the materials. For example, the "to transfer water" function of the pump has a ToProvide meta-function for the "to vaporize" function of the boiler (see mf_1 in Figure 3). If f_t is an attribute function, the function changing the same attribute of the focused material object is said to have a ToProvide meta-function. In general, an object function has another object function as an agent function performing a ToProvide meta-function.

When what f_t focuses on is energy (i.e., f_t is an energy function), the source energy of the output energy is called *material energy*. Such a material energy is transferred to a different medium by f_t or transformed into a different kind of energy. For example, the "to generate heat" function of the burner has a ToProvide meta-function for the "to give heat" function of the boiler (see mf_4), because the combustion gas carries the heat energy. In general, energy functions have the ToProvide meta-function for other energy functions such as mf_4 and mf_{10}.

ToDrive (driver role)

We call such energy that *essentially* causes the internal process of the function f_t as *driving energy*. The necessary conditions for the essentialness are (1)not material of the focused entity of f_t, and (2)intentionally consumed by the process of f_t (see Table 1). The former condition means that the driving energy plays a role not as an object but as an agent for the process of f_t. In the latter condition, what we mean by "intentionally consumed" does not include the unintentional loss such as that during transfer of energy. In many cases, a part of the driving energy is transformed into the different kind of energy and the object focused by f_t brings the transformed energy.

The function which generates or transfers such a driving energy is said to have the meta-function "to drive f_t". For example, the "to give heat" function of the boiler has a ToDrive meta-function for the "to rotate" function of the turbine (see mf_9), because the heat energy is not material of the shaft and is consumed by the rotation. A part of the heat energy is transformed into the kinetic energy which is carried by the focused object (the shaft).

On the other hand, for "to generate kinetic energy", the same function performs not ToDrive but ToProvide (see mf_{10}), because the heat energy is material of the kinetic energy. In general, the ToDrive is a role of an energy function for an object function.

ToEnable (enabler role)

This meta-function is used for representing a mandatory condition playing a crucial role in f_t except ToProvide and ToDrive (see Table 1). What we mean by this weak definition is that the conditions such as the existence of the material and the existence of the driving energy are too obvious to be said to enable the function. ToEnable is primarily a role of an attribute function (see the previous section for the definition) for an object function. On the other hand, ToProvide is a role of an amount function for an object function or a role of an energy function for an energy function. In general, ToDrive is a role of an energy function for an object function.

For example, because the steam of which phase is gas plays a crucial role in occurrence of the heat-expansion process in the turbine and the phase is neither material of rotation nor the consumed energy, the "to vaporize" function of the boiler is said to have a meta-function ToEnable (see mf_5). On the other hand, it performs different meta-function ToProvide for the "to condense" function of the condenser (see mf_6), because the functions focus on the same phase parameter.

ToAllow and ToPrevent

These two meta-functions are concerned with the undesirable side effects of functions. A function f_a having positive effects on the side effect of a function f_{t1} is said to have a meta-function "to allow the side-effects of f_{t1}". The "undesirable side effect" is defined in a relation with another function f_{t2} or the whole system.

If a serious trouble (e.g., faults) is caused in a function f_{t2} when a function f_a is not achieved, the function f_a is said to have a meta-function "to prevent malfunction of f_{t2}". For example, the "super-heat" function of the boiler prevents malfunction of the turbine (mf_7), because the steam of low temperature would damage the turbine blade. In general, almost all f_a performing a ToAllow meta-function for f_{t1} also have a ToPrevent meta-function for the function f_{t2}.

ToImprove and ToEnhance

These meta-functions represent *optional* contribution for f_t. The discrimination between ToImprove and ToEnhance is made by increment of the amount of the input energy. For example, the "to keep low pressure" of the condenser contributes to the efficiency of the "to rotate" function (see mf_{11}) without increment of input energy. On the other hand, the "to super-heat" function of the boiler optionally increases the amount of the input energy (mf_8).

ToControl (controller role)

When a function f_a regularizes the behavior of f_t, its meta-function is said to be "to control f_t". For example, consider a control valve which changes the amount of flow of the combustion gas for the boiler in order to maintain the amount of flow of the steam. It is said to have a meta-function ToControl for the "to vaporize" function of the boiler (not shown in Figure 3).

Although the base-function of the control valve from the aspect of the information flow (see the previous section) is also represented by the same word "to control", the meanings are different. The function type of a macro component which consists of the boiler and the control valve is said to be ToMaintain-type.

Other Examples of Meta-Functions

This section shows examples of meta-functions in other domains, an oil refinery plant and a manufacturing process. Then, we discuss the limitations of domains which the ontology can be applied to.

Oil Refinery Plant (Figure 4)

The meta-functions in an oil refinery to refine crude oil are shown in Figure 4. In the plant, the given crude oil is pre-heated by the heat exchanger, liquefied by the pre-flash drum, vaporized by the heater, and then refined by the fractionator.

The "to vaporize oil" function of the heater is said to *enable* the "to refine" function of the fractionator because the oil of which phase is gas plays a crucial role
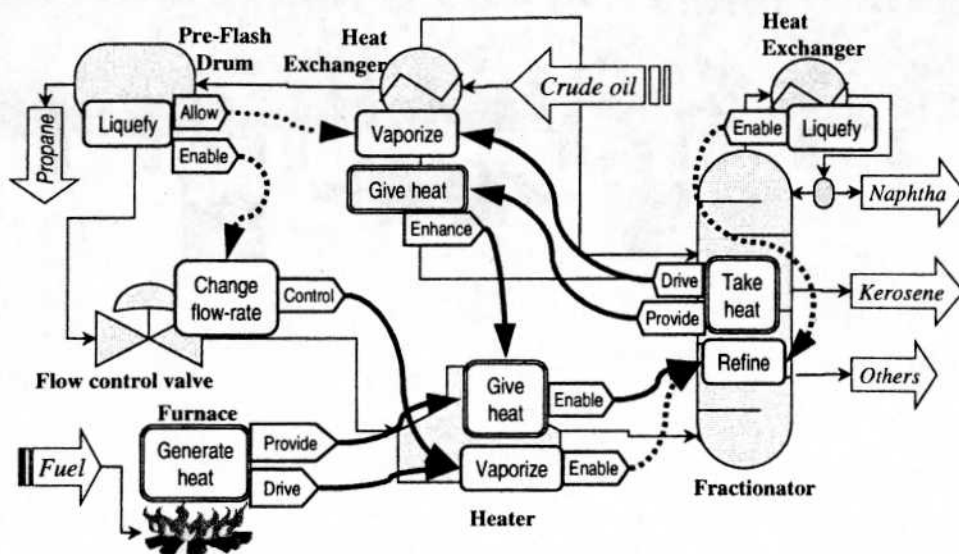
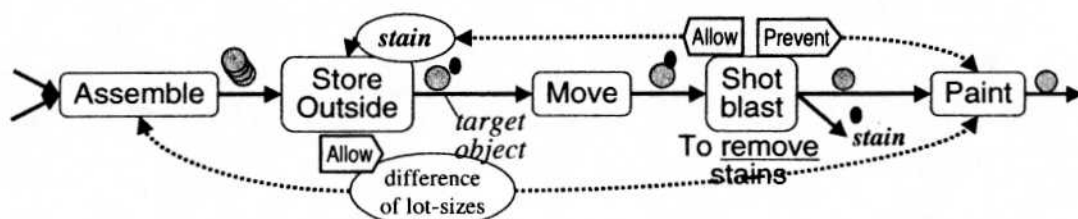Figure 4. Meta-functions in an oil refinery (part)

Figure 5. Meta-functions in a manufacturing process (part)

in occurrence of the process in the fractionator and the phase is neither material of the products nor the consumed energy.

The heat exchanger in which the crude oil is firstly processed exists in order to *enhance* the heater by pre-heating the oil using the heat energy from the fractionator. As an undesirable side effect, a part of the oil is vaporized by the heat exchanger. The pre-flash drum *allows* the vaporization and *enables* the control of the flow-rate by the flow control valve because it is designed to control liquid flow.

Manufacturing process (Figure 5)

Figure 5 shows a part of a manufacturing process and its meta-functions. In the process, target objects are assembled, stored, moved, shot-blasted, and then painted. The reason of the existence of the shot-blasting is to remove stains from the target objects (a base function) which are caused by storing them outside. In other words, the shot blasting process *allows* the target objects to stain in outside and *prevents* failure of painting process caused by stains.

The reason why the objects are stored outside is because it *allows* the big difference between the lot sizes of the assembly and the painting processes. The large space in outside is needed for storing many objects. These meta-functions explicate the reason of the existence of the processes.

Application Domains of the Ontology

Our ontology is also applied to modeling of a concrete chemical plant. The models in the all applications share many functional concepts except those specific to the chemical domain such as "react".

Currently, our ontology adopts the device ontology and assumes the existence of something flowing* (or transferred) among components which carries energy (called objects). Each base-functions is defined as an abstraction of an action of the component on the object using the energy. The meta-functions are mainly defined in terms of the roles played by such objects or energy in the target functions.

It means that we need to capture some discrete structural entities in the target domain and distinguish the agents and the objects in the structural entities. The fluid-related plants such as power plants and chemical plants match the conditions very well. In many manufacturing processes as shown in Figure 5, although they usually do not have the energy-related aspect, we can find devices (or humans) as the agents and the objects processed by the devices. In principle, the electric circuit can be captured by our ontology.

On the other hand, in the mechanical domain, we cannot distinguish the agents and the objects. A mechanical device can be said to perform a function without an object to work on. The ontology currently does not cover mechanical phenomena.

```
1 MetaFuncType identify-meta-function(function f1,component c1,function f2,component c2)
2 begin
3   e1 := focused-entity(f1); e2 := focused-entity(f2);
4   p1 := focused-parameter(f1, e1); p2 := focused-parameter(f2, e2);
5   if exists-causal-relation(p1,p2)
        then fa:=f1;fo=f2;pa:=p1;po:=p2;ea=e1;eo=e2;ca=c1;co=c2;
6   else if exists-causal-relation(p2,p1)
        then fa:=f2;fo=f1;pa:=p2;po:=p1;ea=e2;eo=e1;ca=c2;co=c2;
7   else return null;
8   eoi:=made-of(eo,co);                              // a set of material of eo in co
9   eai:=transitive-same(ea,co,'input);              // the entity propagated by the connection
10  eao:=product-of(ea,co);                          // a set of products of ea in co
11  if not satisfy(simulate(malfunction(fa)), goal(fo)) then // mandatory contribution
12    if member-of(eai, eoi) and                     // for ToProvide
13      (is-a(fo,'amount-f) and is-a(fa,('amount-f)))
14      or (is-a(fo,'attr-f)) and P-Focus(port(eai)) and
            (is-a(fa,'amount-f) or equal(class(p1),class(p2)))
15      or (is-a(fo,'energy-f) and is-a(fa,'energy-f))
16        then return 'ToProvide
17    else if is-energy(ea) and is-object(eo)         / for ToDrive
18      and is-proportional-causality(pa,po) and
19      ((e:=energy-at-port(P-Focus(fo)));member-of(e,eao)
            and (not equal(class(e),class(ea))))
20        or less(amount(at-output(eai,co)),amount(eai)))
21      and (is-a(fa,'generate) or is-a(fa,'shift))
22        then return 'ToDrive
23    else if conditional-causality(pa,po)            // for ToEnable
24        then return 'ToEnable
25    else return null;
26  else // optional contribution
27    ...

Entity focused-entity(function f)
begin
  for e in all entities in c1
    if(has-attr(e,O-Focus(f1)) and (location(e)=P-Focus(f1)) then return e;
end
```

Figure 6: The algorithm to identify meta-functions (part)

Identifying Meta-Functions

Currently, the investigation on a functional understanding system which automatically identifies plausible functional structures from the given structural and behavioral models is in progress on the basis of this functional framework (Kitamura and Mizoguchi, 1998). The meta-functions also can be automatically identified. This section describes the overview of the identification of the whole of functional structure and the details of identification of meta-functions.

Identification of Functional Structures

The problem of identification of functional structures is mapping from the structure and the behavior layers to the base-function and the meta-function layers in Figure 1. The process of identification consists of the following three steps. Firstly, the understanding system exhaustively generates candidates of base-functions to be performed by each component as all tuples of possible values of FTs context-independently. Then, the understanding system screens out meaningless ones by matching them with the concepts in the ontology of the base-function. Such functional interpretations that match with no concept in the ontology are screened out as a meaningless interpretation assuming the completeness of the ontology in the functional space. Although many candidates of the functional interpretations remain, plausible functional interpretations are identified by the following two steps.

Secondly, the understanding system identifies the function types and meta-functions to be performed by each function. Lastly, the hierarchy of the functions are identified according to the knowledge of the way of achievement of functions. See (Kitamura and Mizoguchi, 1998) for the detail of the identification of functional structures except the meta-functions.

Algorithm to Identify Meta-Functions

Given a pair of base-functions, the type of a meta-function between them can be identified according to the algorithm shown in the Figure 6 (part). The identification module has been implemented using Allegro Common Lisp on UNIX workstations.

The relationship among entities such as material-product relations (line 8 and 10) can be identified using the behavioral model. The connection information is provided by the structural model (line 9).

The identification requires the reasoning engine at the behavior layer such as the qualitative reasoning engine. It should answer the questions whether a causal relation exists between the focused parameters (line 5 and 6) and whether the goal of the target function is satisfied when the agent function does not work (line 11). We use our original qualitative reasoning engine based on the behavioral model representing the normal behavior of the target system (Kitamura et al., 1996; 1997).

In order to identify the ToPrevent meta-functions, the knowledge of unintended abnormal phenomena is also needed. Our original diagnostic reasoning engine based on general fault models is used in our implementation for this purpose (Kitamura and Mizoguchi, 1999).

The functional system in terms of O-Focus and P-Focus enable the understanding module to identify the focused entities (line 3) and the focused parameters (line 4).

Example of Steps of Identification

Imagine that we are given "to generate heat" (denoted by f1 in Figure 6) of the furnace (c1) and "to vaporize water" function (f2) of the boiler (c2) in Figure 3. The focused entity of the furnace (e1) is the heat energy of the combustion gas and its focused parameter (p1) is the amount of the heat energy. On the other hand, the focused entity of the boiler (e2) is the steam and the focused parameter (p2) is its amount. Because there is a causal relation from p1 to p2, then the heat generation is the agent function and the vaporization is the target function. The material of the focused entity of the target function (denoted by eoi) is the inlet water (line 8).

Because the boiler does not work without the heat-generation function, the meta-function is mandatory (line 11). Because the focused entity of the furnace (the heat energy of the combustion gas) is not member of the eoi (the inlet water), the meta-function is not ToProvide (line 12).

Next, the conditions for ToDrive are checked. Firstly, the focused entity of the agent function (e1, the heat energy) is an energy and the focused entity of the target function (e2, the steam) is an object (line 17). Secondly, the causal relation between the amount of the heat energy and the amount of the steam is basically proportional (line 18). Thirdly, the amount of the heat energy is reduced by the boiler (line 20). Lastly, the focused attribute of the agent function (p1) is the amount (line 21). Then, the meta-function is ToDrive.

Utility of the Functional Ontology and Meta-Functions

Generation of Explanation

We have developed an explanation generation system based on this functional framework (partly discussed in (Sasajima et al., 1995)). It has been successfully applied to a simple model of a power plant shown in Figure 3 and a concrete chemical plant different from that shown Figure 4. The explanation of the latter is verified by domain experts.

The meta-functions are used for explaining rationales of organization of the target system from the viewpoint of the system engineering without mention of changes of entities specific to the target system, that is, what types of collaboration with other functions are done by the functions in order to make the whole system work.

It can provide explanations at several abstraction levels in terms of well-defined sharable functional concepts and thus facilitate to share the understanding of the target systems by human designers.

Redesign

Currently, the investigation on a redesign system to propose improvements of existing artifacts is in progress on the basis of this functional framework (Kitamura et al., 1998). In general, functional representation enables the (re)design system to reason at the functional level as discussed in a rich literature (e.g., (Bradshaw and Young, 1991; Goel and Chandrasekaran, 1989; Hodges, 1992; Bhatta and Goel, 1997)). The ontology provides us a comprehensive vocabulary for design knowledge and specifies the reasoning space. For example, as discussed in (Hodges, 1992), the design system can select a component suitable to realize a function from the component library indexed in terms of the systematized functional concepts in the ontology.

The meta-functions enable us to capture the dependency among functions at an abstraction level. As a way of the dependency management (Lee, 1997), the model of dependency among functions in terms of meta-functions enables the redesign system to propose drastic improvements of an existing artifact with consideration of such dependency in the original design.

Identification of functional structures

As discussed in the previous section, the functional ontology enables us to identify the functional structures from the behavioral and structural models automatically (Kitamura and Mizoguchi 1998). Although such task in principle difficult because the search space of function is huge, the ontology plays a role to limit the search space. The ontology provides such primitives in the functional space that are targets in the mapping, and screens out meaningless functional interpretations.

The identification (or description) of the meta-functions contributes to identifying the functional hierarchies. As pointed out in (Keuneke, 1991; Snooke and Price 1997), the identity of the component from the viewpoint of function in the functional hierarchies is different from that from the structural (or topological) viewpoint. Then, the identification of functional groups of the given structural components is one of the crucial issues in order to identify the functional hierarchies. Some heuristics for specifying conditions (or preferences) for grouping functions have been identified (Kitamura and Mizoguchi 1998).

Because each types of meta-functions has own strength to make the functional groups, the grouping of given functions can be done according to the types of the meta-functions among them. An investigation on the grouping heuristics according to the types of the meta-functions is now in progress.

Related Work

The functions in (de Kleer, 1984; Umeda et al., 1990; Lind, 1994; Sasajima et al., 1995; Chandrasekaran and Josephson, 1996) represent abstracted behavior of components and are defined as base-functions in our framework. The meta-function represents a role for another function without mention of changes of incoming or outgoing objects of components and hence is totally different from such base functions. In (Sembugamoorthy and Chandrasekaran, 1986; Vescovi et al., 1993), function is defined as a kind of hierarchical abstraction of behavior. It corresponds to the is-achieved-by relations among functions in our framework.

The CPD in CFRL (Vescovi et al., 1993) represents causal relations among functions which correspond to relation among functions on the function layer in our framework. Lind categorizes such relations into Connection, Condition and Achieve (Lind, 1994). Rieger identifies "enablement" as a type of the causal relation between states and action (Rieger and Grinberg, 1977). The meta-functions are results of interpretation of such causal relations between functions under the role of the agent function for the target functions.

As types of base-functions, we redefine ToMake and ToMaintain (Keuneke, 1991). Our other functional type "ToHold" is similar to "ToAllow" identified in (Bonnet, 1992).

In (Hodges, 1992; Borst et al., 1997), the sets of "primitives of behavior" are proposed. Lind identifies a few general functions such as "storage of energy" which are categorized into multiple levels (Lind, 1994). We added more intention-rich concepts such as "remove" with unnecessary intention to the set of the base functions and organized in is-a and part-of hierarchy. Furthermore, we identify some types of the meta-functions.

In Value Engineering research (Miles, 1961), standard sets of verbs (i.e., functional concepts) for value analysis of artifacts are proposed (Tejima et al., 1981). It enables the human designers to share descriptions of functions of the target artifacts. However, they are designed only for humans, and there is no machine understandable definition of concepts.

The consolidation theory (Bylander and Chandrasekaran, 1985) tries to capture the general patterns of aggregation of the system. In a literature on design, many general "patterns" of synthesis are proposed (e.g., (Bradshaw and Young, 1991; Bhatta and Goel, 1997)). Our general ways of achievement, however, explicitly represent the feature of achievement such as theory and phenomena. The importance of such key concepts in design is pointed out in (Takeda et al., 1990). They enable the redesign system to facilitate the smooth interaction between models at the structural and functional levels.

The idea of the meta-functions, the meta-layer, and the ways of achievement in our functional framework are not shown in our previous papers. FBRL reported in (Sasajima et al., 1995) represents only the base-functions. The paper (Kitamura and Mizoguchi, 1998) reported a basic idea of an ontology of functional concepts without the meta-functions and the meta-layer and discussed its utility in the functional understanding task. The functional decomposition patterns shown in that paper do not include the explicit representation of *ways of achievement* discussed in this paper.

Limitations

Completeness and Naturalness. We do not claim completeness of the set of concepts and the appropriateness of the *labels* of the concepts shown in this paper. Note that we define precisely the meaning of concepts for discrimination. The definitions may narrow than those we use in the natural language, because we tend to use them confusedly.

Meta-functions in the achievement relations. The meta-functions discussed in this paper are concerned with dependency among functions of the same grain-size. That among functions of the achievement relations in the different grain-size is under consideration.

Summary

We proposed nine types of the meta-functions, which represent the types of collaboration and dependency among functions of components. They contribute to explanation of design rationale as a system and redesign with consideration of such dependency. We also discussed an ontology of functional concepts including the meta-functions, aiming at a comprehensive vocabulary for functional representation.

Although our ontology is not formally defined in terms of logic such as Ontolingua (Gruber, 1993), we have defined clearly the meaning of functional concepts using the mapping primitives to the behavior called FTs. Such definitions enable a redesign supporting system to facilitate the smooth interaction between models at the structural and functional levels. We are currently designing a language named FBRL II supported by this functional ontology.

Acknowledgments

References

Abu-Hanna, A., and Jansweijer, W. 1994. Modeling domain knowledge using explicit conceptualization, IEEE Expert, 9(5):53-63.

Bhatta, S. R., and Goel, A. K. 1997. A functional theory of design patterns. In Proc. of IJCAI-97, 294-300.

Bonnet, J. C. 1992. Towards a formal representation of device functionality. TR 92-54, Knowledge Systems Laboratory, Stanford University.

Borst, P.; Akkermans, H.; and Top, J. 1997. Engineering ontologies. Int. J. Human-Computer Studies 46:365-406.

Bradshaw, J. A., and Young, R. M. 1991. Evaluating design using knowledge of purpose and knowledge of structure. IEEE Expert 6(2):33-40.

Bylander, T., and Chandrasekaran, B. 1985. Understanding behavior using consolidation, Proc. of IJCAI-85, 450-454.

Chandrasekaran, B.; Goel, A. K.; and Iwasaki, Y. 1993. Functional representation as design rationale. COMPUTER, 48-56.

Chandrasekaran, B., and Josephson, J. R. 1996. Representing function as effect: assigning functions to objects in context and out. In Proc. of AAAI-96 Workshop on Modeling and Reasoning with Function.

de Kleer, J. 1984. How circuits work, Artificial Intelligence 24:205-280.

Goel, A., and Chandrasekaran, B. 1989. Functional Representation of Designs and Redesign Problem Solving. Proc. of IJCAI-89, 1388-1394.

Gruber, T. R. 1993. A translation approach to portable ontology specifications, Knowledge Acquisition, 5(2):199-220.

Hodges, J. 1992. Naive mechanics - a computational model of device use and function in design improvisation. IEEE Expert 7(1):14-27.

Keuneke, A. M. 1991. A. device representation: the significance of functional knowledge. IEEE Expert, 24:22-25.

Kitamura, Y.; Ikeda, M.; and Mizoguchi, R. 1996. A Qualitative Reasoning based on an Ontology of Fluid Systems and Its Evaluation on a Power Plant. Proc. of PRICAI'96, pp.288-299, Springer-Verlag.

Kitamura, Y.; Ikeda, M.; and Mizoguchi, R. 1997. A Causal Time Ontology for Qualitative Reasoning, Proc. of IJCAI-97, pp.501-506

Kitamura, Y., and Mizoguchi, R. 1998. Functional ontology for functional understanding, Papers of Twelfth International Workshop on Qualitative Reasoning (QR-98), 77-87.

Kitamura, Y.; Ueda, T.; Fuse, M.; and Mizoguchi, R. 1998. Towards Redesign of Manufacturing Processes based on Functional Understanding, Poster proceedings of AI in Design 98.

Kitamura, Y., and Mizoguchi, R. 1999. An Ontological Analysis of Fault Process and Category of Faults, Working papers of DX-99, forthcoming.

Lind, M. 1994. Modeling goals and functions of complex industrial plants. Applied Artificial Intelligence, 8:259-283.

Lee, J. 1997. Design rationale systems: understanding the issues. IEEE Expert, 12(3):78-85.

Miles, L. D. 1961. Techniques of value analysis and engineering. McGraw-hill.

Mizoguchi, R., and Ikeda, M. 1997. Towards ontology engineering. In Proc. of PACES/SPICIS '97, 259-266.

Rieger, C., and Grinberg, M. 1977. Declarative representation and procedural simulation of causality in physical mechanisms. In Proc. of IJCAI-77, 250-256.

Sasajima, M.; Kitamura, Y.; Ikeda, M.; and Mizoguchi, R. 1995. FBRL: A Function and Behavior Representation Language. In Proc. of IJCAI-95, 1830-1836.

Sembugamoorthy, V., and Chandrasekaran, B. 1986. Functional representation of devices and compilation of diagnostic problem-solving systems. In Experience, memory and Reasoning, 47-73.

Snooke, N., and Price, C. 1997. Hierarchical Functional Reasoning. In Proc. of IJCAI-97 Workshop on Modeling and Reasoning about Function, 11-22.

Takeda, H.; Veerkamp, P.; Tomiyama, T.; and Yoshikawa, H. 1990. Modeling design processes. AI Magazine, 11(4), 37-48.

Umeda, Y. et al., 1990. Function, behavior, and structure. AI in Engineering, 177-193, 1990.

Tejima, N. et al. eds. 1981. Selection of functional terms and categorization. Report 49, Soc. of Japanese Value Engineering (In Japanese).

Vescovi, M.; Iwasaki, Y.; Fikes, R.; and Chandrasekaran, B. 1993. CFRL: A language for specifying the causal functionality of engineered devices. In Proc. of AAAI-93, 626-633.