FBS Modeling: Modeling Scheme of Function for Conceptual Design

Yasushi Umeda and Tetsuo Tomiyama

Graduate School of Engineering, The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan
phone +81-3-3812-2111 (ext. 6481) fax +81-3-3812-8849
email {umeda, tomiyama}@zzz.pe.u-tokyo.ac.jp

Abstract: Function is gradually recognized important within the AI community, since this is the bridge between human intention and physical behavior of artifacts particularly at conceptual design stage. However, to support conceptual design, there are two research topics; namely, knowledge representation scheme that allows a user to describe functions flexibly without reference to behavior and structure and a CAD system that supports the design process in which functions are gradually embodied into behavioral and structural models. This paper proposes a new knowledge representational scheme for functions, called *Function-Behavior-State (FBS) modeling*, that defines a function as an association of human intention and behavior and represents a design object hierarchically. For clarifying advantages of the FBS modeling, we describe a computer tool, called an *FBS Modeler*, that supports the conceptual design. We also formalize design processes with this modeler and describe an innovative application that increases reliability of design objects by using functional knowledge.

1 Introduction

While qualitative physics enables us to deal with knowledge about behavior, it is still difficult to represent and manipulate knowledge about function on a computer. Since function is the bridge between human intention and physical behavior of artifacts, this issue is critical for developing knowledge based systems for applications such as design and diagnosis. For example, in the early stage of a design process (*i.e.*, conceptual or functional design), a designer should determine the functional structure of a design object and its basic mechanisms that affect performance and quality of the design object definitely. Therefore, it is crucial to develop methodologies representing and manipulating functions for constructing CAD systems for functional design.

There is no clear and uniform definition of *function* and, moreover, it seems impossible to describe function objectively. This is mainly because function is an intuitive concept depending on intentions of designers or users. Rodenacker [1] defines a function as a relationship between input and output of energy, material, and information and this definition is widely accepted in design research (*e.g.* [2, 3]).

Value engineering (VE) represents function in the form of *to do something* [4]. Although this definition is general, the representation is not powerful enough for design because of difficulties for implementations. Within the AI community, research that deals with functions increases (e.g., [5, 6, 7, 8]). They use functional knowledge mainly for analytical tasks such as evaluation of simulation and diagnosis. In other words, they derive functions from behavioral and structural models. In contrast, for supporting functional design, behavioral and structural models should be constructed from functions.

Therefore, the following two research issues still remain for building a CAD system that supports the conceptual design:

- Knowledge representation scheme that allows the user to describe functions flexibly without reference to behavior and structure.
- A CAD system that should support the design process in which the required functions are gradually embodied into behavioral and structural models by, e.g., decomposing functions.

In this paper, first we propose a new knowledge representation scheme for functions called *Function-Behavior-State (FBS) modeling* that takes the above two issues into consideration. Then, we demonstrate the importance and advantages of the FBS modeling by illustrating a computer tool to support the conceptual design based on the FBS modeling. Finally, we discuss advantages and issues of the FBS modeling compared with other approaches.

2 FBS Modeling

2.1 Representation of Function

In the FBS modeling, we define a function as follows:

Definition 1 *function*

A function is "a description of behavior recognized by a human through abstraction in order to utilize it." Therefore, function f is represented as a tuple (f_{symbol}, b) where f_{symbol} and b denote a symbol in the form of *to do something* and behavior that can realize this function, respectively.

As Definition 1 shows, it is difficult to clearly distinguish function from behavior and it is not meaningful to represent function independently of the behavior from which it is abstracted. Therefore, we represent a function by its symbol representing the designer's intention as Value Engineering proposed and its physical semantics (*behavior*). Although the symbol is meaningful only to a human, this information, associated with its behavior, is essential for supporting design such as reuse of design results and clarification of specifications.

The relationship between functions and behaviors in Definition 1 is subjective and many-to-many correspondent; for example, we might say that behaviors such as "collision of two objects" and "oscillation of a string" are used for a function "to make a sound." We call this relationship *F-B relationship*.

Definition 2 *F-B relationship*
An F-B relationship is a many-to-many correspondence represented by the following Boolean function f_{f-b}, where F is a set of functions:

$$f_{f-b}(f_{symbol}, b) = \begin{cases} true, & \text{if } (f_{symbol}, b) \in F \\ false, & \text{otherwise} \end{cases} \tag{1}$$

We assume that the representation of function includes subjective intentions, whereas the representation of behavior can be determined more objectively based on physical principles. Here, we first define states, behaviors, and then physical phenomena.

Definition 3 *state*[1]
A state is defined by a triplet (E, A, R) where:

E	:	identifiers of entities included in this state.
A	:	attributes of entities
R	:	relations in this state that include relations among entities, between entities and attributes, and among attributes.

Definition 4 *behavior*
A behavior is defined by sequential one and more changes of states over time and represented by the following ordered list, where S and T denote a set of states and an ordered set of time, respectively:

$$b = (s_0, t_0), (s_1, t_1), \ldots, (s_n, t_n) \ (n \geq 0, s_i \in S, t_i \in T) \tag{2}$$

Definition 5 *physical phenomena*
A physical phenomenon *pp* causes a state transition from (s_i, t_i) to (s_j, t_j) $(i \leq j)$ and is represented as follows:

$$pp = (s_i, \Delta s, \Delta t) \ \ (\Delta s = s_j - s_i, s_i, s_j \in S, \\ \Delta t = t_j - t_i, t_i, t_j \in T, i \leq j) \tag{3}$$

Where, s_i represents the required condition for activating this phenomenon.

According to the assumption above, behavior b can be described by its initial state (s_0, t_0) and a set of physical phenomena PP. We here represent behavior by introducing a mathematical function Sim as follows. The function Sim corresponds to behavior simulation. In our implementation described in Section 3, this is carried out by QPT (Qualitative Process Theory) [9].

$$b = Sim((s_0, t_0), PP) \tag{4}$$

However, representations of behavior may differ depending on the physical situations of the current interest. For example, while the behavior that electricity passes through a wire can be codified by resistance, voltage, current, and so on, it can also be captured as motion of electrons. To represent this difference, we introduce *aspects*.

Definition 6 *aspect*
An aspect ASP is defined as follows, where E^0, A^0, R^0, PP^0, and T^0 denote sets of all entities, attributes, relations, physical phenomena and time of the current interest, respectively:

$$ASP = (E^0, A^0, R^0, PP^0, T^0) \tag{5}$$

Figure 1 illustrates the relationship among function, behavior, state, and aspect described above and suggests how a function can be described. That is, To describe a function f, the user should describe its symbol f_{symbol} and behavior b to realize it. Then, to describe a behavior b, the user should select an appropriate aspect ASP and describe its initial state (s_0, t_0) and physical phenomena.

2.2 FBS Diagram

In design, the designer decomposes the required functions into subfunctions hierarchically until arriving at substantial components (e.g., [2]). Therefore, it is essential for supporting design to represent a design object hierarchically in such a way that its representation becomes gradually concrete over the hierarchy. We assume that such hierarchies can only be constructed subjectively from the viewpoint of function rather than objectively or physically.

Definition 7 *functional decomposition*
A designer can decompose function f into subfunctions and, conversely, recognize a function from a set

[1]In this paper, we call so-called *structure* and so-called *state* altogether *state*, because the distinction between them is not essential. For example, even structure of a machine might change when a fault occurs.

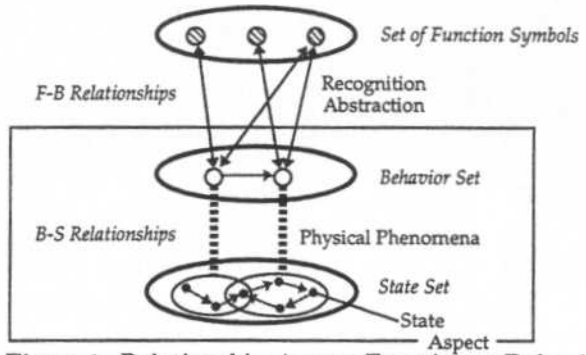

Figure 1: Relationship Among Function!$ Behavior!$ State, and Aspect

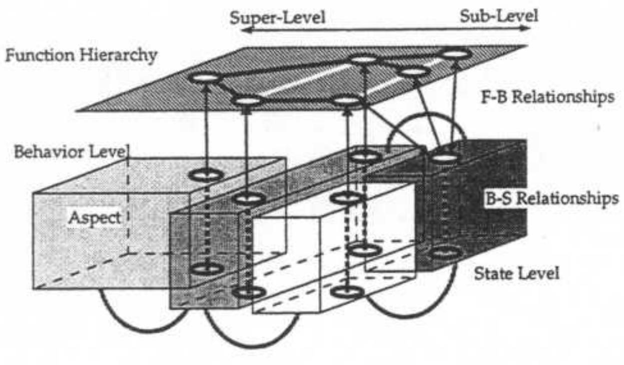

Figure 2: FBS Diagram

of subfunctions by some means. This ability is represented by the following Boolean function f_{decomp}, where g_f denotes a graph of subfunctions corresponding to f, and F_{sub} and R_{ff} represent subfunctions and relations among subfunctions such as conjunction and temporal order among subfunctions in this graph, respectively:

$$f_{decomp}(f, g_f) = \begin{cases} true, \\ \text{(if } f \text{ can be decomposed into } g_f) \\ false, \\ \text{(otherwise)} \end{cases} \tag{6}$$

$$g_f = (F_{sub}, R_{ff}) \tag{7}$$

As a result of functional decompositions, a design object is represented hierarchically as shown in Figure 2. We call this scheme a *Function-Behavior-State (FBS) diagram.* This figure indicates that a designer should concurrently describe symbols of functions and their realizing behaviors with appropriate aspects hierarchically. Since aspects are physically related with each other, consistency of behaviors and states can be maintained if a sufficient amount of physical knowledge is collected. For example, Kiriyama [10] proposes the *metamodel* mechanism that manages relationships among aspects.

The main ideas of the FBS diagram are:

- to distinguish subjective part of a design object (*function symbols*) and objective part (*behaviors* and *states*)
- to represent a function as an association of subjective concepts (*function symbols*) and objective concepts (*behaviors*) rather than just either of them; *i.e.*, functions relate subjective concepts and objective concepts
- to represent a design object hierarchically in order to support a modeling process that details functional and behavioral descriptions concurrently.

Owning to these features, computerization of the FBS diagram helps a designer to execute functional design; for instance, the system can search for appropriate behaviors to a required function, check inconsistencies of the objective part, and propose modification for the inconsistencies.

2.3 Causal Decomposition and Task Decomposition

We here propose that decompositions of functions can be classified into the following two categories. Here, a function f_0 is decomposed into subfunctions f_1, f_2, ..., f_n that are embodied by behaviors b_1, b_2, ..., b_n.

Causal decomposition For instance, function f_0 "to generate light" can be decomposed as follows:

f_1 : to light lamp with electricity
b_1 : lamp lighting
f_2 : to generate electricity
b_2 : battery generating electricity

In this case, b_2 is indispensable for activating b_1. We call this kind of decomposition *causal decomposition.*

Definition 8 *causal decomposition*
Causal decomposition is a decomposition that behaviors resulting from this decomposition are causally related with each other and satisfies the following formula, where predicate $occur(b)$ denotes that behavior b can occur and $b_1 + b_2$ represents an operation to connect b_1 and b_2 appropriately:

$$\exists b_0 : f_{decomp}(f_0, \{f_1, f_2\}) = true \land f_1 = (f^1_{symbol}, b_1) \land f_2 = (f^2_{symbol}, b_2) \land f_0 = (f^0_{symbol}, b_0) \land b_0 = b_1 + b_2 \land (\neg occur(b_1) \lor \neg occur(b_2)) \land occur(b_0) \tag{8}$$

Task decomposition For example, a function f_0 of a CD player "to scan a CD" can be decomposed as follows:

f_1 : to rotate the disk
b_1 : rotation of a motor
f_2 : to read the data
b_2 : laser lighting

This kind of decompositions is task decomposition rather than causal decomposition. Namely, b_1 and b_2 are not causally related with each other, since they can occur independently.

Definition 9 *task decomposition*
Task decomposition is a decomposition that satisfies the following formula:

$$\exists b_0 : f_{decomp}(f_0, \{f_1, f_2\}) = true \land f_1 = (f^1_{symbol}, b_1) \land f_2 = (f^2_{symbol}, b_2) \land f_0 = (f^0_{symbol}, b_0) \land b_0 = b_1 + b_2 \land occur(b_1) \land occur(b_2) \land occur(b_0) \tag{9}$$

In the implementation described in Section 3, while task decompositions are described as knowledge, causal decompositions are executed by the system by using physical knowledge.

3 Implementation: the FBS Modeler

We have developed a tool called an *FBS Modeler* to support functional design based on the FBS diagram. Because of implementational limitations, the modeler can handle only one *aspect* at one time that represents

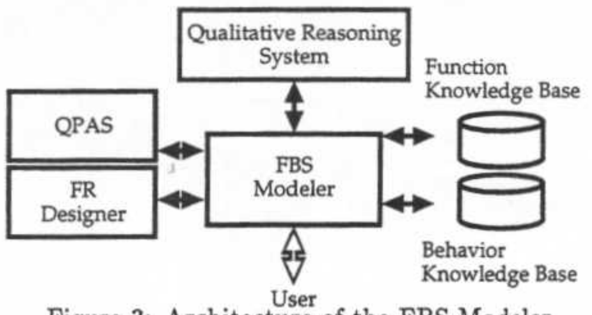

Figure 3: Architecture of the FBS Modeler

behaviors at the bottom level in Figure 2. Functions in the modeler are represented as follows:

$$f = \begin{cases} (f_{symbol}, b), & \text{if } f \text{ is a bottom function} \\ f_{symbol}, & \text{otherwise} \end{cases} \tag{10}$$

Figure 3 shows the architecture of the FBS Modeler. The qualitative reasoning system based on QPT gives representational scheme of behaviors and states and executes the behavioral simulation [10]. QPAS (Qualitative Process Abduction System) [11] that derives physical phenomena that realize the given state transitions is incorporated for supporting the causal decomposition. The FR Designer will be described in Section 3.3. Knowledge about function is described in the function knowledge base in the form of function prototypes described in Section 3.1. The behavior knowledge base consists of physical features, physical phenomena, relations, and entities described in Section 3.1.

3.1 Knowledge Representation

Functional Knowledge

As discussed in Section 2, it is impossible to give the Boolean functions f_{f-b} in Equation (1) and f_{decomp} in Equation (6) in a universal manner. Therefore, we describe them by collecting prototypes of functions from existing design results. We call this kind of knowledge *function prototypes*. A set of all function prototypes F_{proto} is stored in the function knowledge base of the FBS Modeler.

Definition 10 *function prototype*
A function prototype is a triplet $(f_{symbol}, PF_{proto}, G_{proto})$ where:
$PF_{proto} = \{pf | pf \in PF, f_{f-b}(f_{symbol}, pf) = true\}$
PF : a set of all physical features
(to be described in Section 3.1)
$G_{proto} = \{g_f | f_{decomp}(f_{symbol}, g_f) = true\}$

We call PF_{proto} and G_{proto} *F-B knowledge* and *decomposition knowledge* of the function prototype, respectively. Here, as described in Section 2.3, only task decompositions are included in the decomposition knowledge.

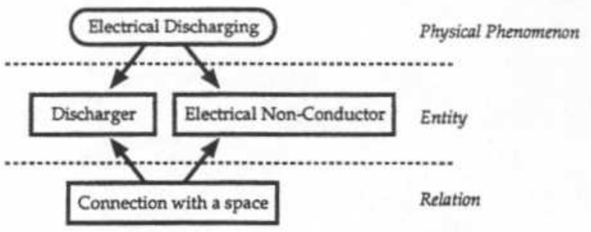

Figure 4: Example of Physical Feature

Physical Feature

In the FBS Modeler, behaviors and states are represented based on QPT. In the context of design, we modified QPT to include three kinds of concepts [10]; namely, *physical phenomena*, *entities*, and *relations*. Physical phenomena correspond to processes and individual views in QPT and entities correspond to individuals. Relations that represent structural relations among entities such as "connected," are represented as preconditions in the processes and the individual views in QPT. One of the advantages of this modification is that structure of the model (*i.e.* entities and relations) becomes explicit.

The F-B relationship described in Section 2.1 implies that it is useful to collect building blocks of behaviors that correspond to function prototypes. To represent such knowledge, Kiriyama proposed *physical features* that are building blocks consisting of components and physical phenomena occurring on the components [10]. We introduce the physical feature to relate functions to behaviors.

Definition 11 *physical feature*
A physical feature pf is a building block of behavior corresponding to a function prototype and is represented as follows, where E, R, PP denote sets of all entities, relations, and physical phenomena, respectively:

$$pf = (E^{pf}, R^{pf}, PP^{pf})$$
$$(E^{pf} \subset E, R^{pf} \subset R, PP^{pf} \subset PP) \tag{11}$$

The behavior knowledge base in Figure 3 consists of the sets of entities, relations, physical phenomena, and physical features in the aspect.

Figure 4 depicts an example of the physical feature in which a discharger is discharging. And Table 1 defines the phenomenon *electrical discharging* that occurs among *discharger* and *electrical non-conductor* that are in relation of *connection with a space.*

FBS Model

We call a model of the design object an *FBS model.*

Definition 12 *FBS model*
An FBS model m_{fbs} is a set of functions defined as follows, where predicates $proto(f_{symbol})$, $PF(fp)$, and $G(fp)$ denote a function prototype of which symbol is f_{symbol}, physical features and the graphs of subfunctions included in a function prototype fp, respectively:

Table 1: Definition of Electrical Discharging

Item	Subitem	Contents
Phenomenon		Electrical Discharging
Supers		Fundamental

Conditions	prerequisites	discharger=Discharger
		device=Electrical Non-Conductor

	references	
	relations	connection with a space(discharger,device)
	q-conditions	(discharger voltage) > zero
	s-conditions	(discharger sw)=on
Influences	quantities	(device charge) = (minus @zero plus)
	q-relations	(device charge) direct (discharger voltage)
	influences	

$$m_{fbs} = \{f \mid f = (f_{symbol}, g_f) \vee (f_{symbol}, pf) \vee f_{symbol}, proto(f_{symbol}) \in F_{proto}, pf \in PF(proto(f_{symbol})), g_f \in G(proto(f_{symbol}))\} \quad (12)$$

3.2 Design Process with the FBS Modeler

We view that functional design of an artifact with the FBS Modeler is to construct a consistent and feasible FBS model of the artifact by detailing and embodying one or more required functions. We model functional design process as follows.

1. Specification of required functions

   First, the designer selects required functions from function prototypes and adds them to the model. This operation is represented by Equation (13), where $\rightarrow$ denotes an operation of the designer.

$$m_{fbs} = m^0_{fbs} \rightarrow m_{fbs} = m^0_{fbs} \cup \{f_{symbol}\} \quad (proto(f_{symbol}) \in F_{proto}) \quad (13)$$

2. Functional Decomposition

   The designer recursively decomposes the required functions into subfunctions by applying the decomposition knowledge of function prototypes. These decompositions are task decomposition. Equation (14) denotes the operation in which function f_0 is decomposed. As a result, the designer constructs a function hierarchy.

$$m_{fbs} = m^0_{fbs} \cup \{f_0\} \rightarrow m_{fbs} = m^0_{fbs} \cup \{(f_0, g_f)\} \cup F' \quad (g_f \in G(proto(f_0)), F' = \{f \mid f = f_{symbol} \in g_f\}) \quad (14)$$

3. Embodiment

   The designer instantiates physical features that can embody the hierarchy by using the F-B knowledge of function prototypes for each undecomposable function. The operation to embody function f_0 is depicted by;

$$m_{fbs} = m^0_{fbs} \cup \{f_0\} \rightarrow m_{fbs} = m^0_{fbs} \cup \{(f_0, pf)\} \quad (pf \in PF(proto(f_0))) \quad (15)$$

4. Causal decomposition

   After instantiating physical features, the designer often finds out that some of them cannot occur by themselves. In such a case, QPAS supports the designer to instantiate additional physical features [11]. After adding a physical feature, the FBS Modeler assists the designer to add a function corresponding to the added feature by searching through function prototypes in order to explain why the feature is added. This process corresponds to the causal decomposition described in Section 2.3 and, therefore, we do not describe the causal decompositions in the decomposition knowledge. This process in which an additional feature pf_a is added to a target feature pf_0 is represented as follows, where predicate $causedBy(f_1, f_2)$ denotes a relation meaning that function f_1 is caused by function f_2:

$$\begin{aligned}
&m_{fbs} = m^0_{fbs} \cup \{(f_0, pf_0), (f_1, gf_1)\} \quad (f_0 \in gf_1) \\
&\rightarrow m_{fbs} = m^0_{fbs} \cup \{(f_0, pf_0), (f_1, gf_1), pf_a\} \\
&\quad (\neg occur(pf_0) \wedge occur(pf_0 + pf_a)) \\
&\rightarrow m_{fbs} = m^0_{fbs} \cup \{(f_0, pf_0), (f_1, gf_1), (f_a, pf_a)\} \\
&\quad (pf_a \in PF(proto(f_a))) \\
&\rightarrow m_{fbs} = m^0_{fbs} \cup \{(f_0, pf_0), (f_1, gf'_1), (f_a, pf_a)\} \\
&\quad (gf'_1 = gf_1 \cup \{(f_a, causedBy(f_0, f_a))\}) \quad (16)
\end{aligned}$$

5. Construction of the behavior network

   Next, the designer should construct a physically consistent network (called a *behavior network*) by connecting instantiated physical features so as to complete the functional hierarchy. This is done by unifying the same entities in different features. Equation (17) represents this operation in which entities e_1 and e_2 are unified, where E_{fbs} denotes all entities included in this model:

$$E_{fbs} = E^0_{fbs} \cup \{e_1, e_2\} \rightarrow E_{fbs} = E^0_{fbs} \cup \{e'\} \quad (e_1, e_2, e' \in E) \quad (17)$$

6. Evaluation

After constructing the behavior network, the qualitative reasoning system executes behavior simulation. The reasoning system indicates the following information by comparing the initial FBS model m_{fbs} with the result of simulation b_{sim}.

Unrealizable Phenomena: If physical phenomena designated by the designer do not occur in the simulated network, some conditions should be inadequate. An unrearizable phenomenon pp_{not} satisfies the following formula:

$$pp_{not} \in m_{fbs} \land \neg(pp_{not} \in b_{sim}) \qquad (18)$$

Side-Effects: Phenomena that are not expected to occur may cause side-effects that the designer did not notice. A phenomenon pp_{side} which satisfies the following formula is a side-effect:

$$\neg(pp_{side} \in m_{fbs}) \land pp_{side} \in b_{sim} \qquad (19)$$

Unrealizable functions: If functions have unrealizable phenomena in their F-B relationships or unrealizable subfunctions, they will not be realized. Unrealizable functions F_{not} are defined by:

$$F_{not} = \{f|(f = (f_{symbol}, pf) \land pp_{not} \in pf) \lor (f = (f_{symbol}, gf) \land gf \cap F_{not} \neq \phi)\} \qquad (20)$$

Unless satisfied with the result of evaluation, the designer repeatedly refines the function hierarchy and/or the behavior network.

Figure 5 shows an example of the FBS model after the evaluation of a required function "to charge drum," which often appears in design of photocopiers. This model is constructed by decomposing the function into two subfunctions and embodying the subfunctions. The function "to discharge electricity to drum" is embodied by the physical feature shown in Figure 4 and the entities *Drum* and *Shaft* are unified. This figure shows that the required function cannot be realized because the phenomenon *Electric-discharging* is unrealizable because of lack of voltage of *Discharger* (see Table 1).

3.3 Design for Function Redundancy

To demonstrate a practical advantage of the FBS modeling, we illustrate an application of the FBS Modeler; *i.e.*, design for *function redundancy.*

Function redundancy is a design strategy to increase reliability of the design object by using potential functions of existing parts in a slightly different way from the original design [12]. For example, in case of emergency, a car with a manual transmission can run for a while with its starting motor. In such a case, the starting motor exhibits a redundant function of driving the car besides starting the engine.

Definition 13 *function redundancy*
A redundant function is a function that can be realized by other physical features than the feature that realizes the function in its normal state and, therefore, satisfies the following formula. Here, we define a function redundancy as a tuple (f^R_{symbol}, pf').

$$\exists pf' \in m_{fbs} : (f^R_{symbol}, pf_0) \in m_{fbs} \land pf_0 \neq pf' \land f_{f-b}(f^R_{symbol}, pf') = true \qquad (21)$$

The FBS Modeler assists design for function redundancy. The modeler is indispensable because it represents the F-B relationships and supports design. We developed a system called *Function Redundancy (FR) Designer* (see Figure 3). The FR Designer derives candidates of functional redundancies that satisfy Equation (21) partially or totally. Here, the function f_{f-b} is substituted by the F-B knowledge of function prototypes and *partially* means that feature pf' in Equation (21) is partially included in the model and, therefore, the function redundancy can be realized by adding some entities and/or relations to the model.

Based on this idea, we have developed a functionally redundant copier [12].

3.4 Experimental Use of the FBS Modeler

In order to evaluate the FBS Modeler, we asked some designers to use the modeler. They need about a week to learn the modeler. As a result, the following advantages are clarified: First, the FBS Modeler is useful for functional design not only of mechanical design but also of house design. This advantage is owning to the capability of the modeler to represent abstract concepts, which could not be represented with traditional CAD systems, in the form of functions and qualitative physics. Such knowledge is useful not only for novice designers but also for experts because it clarifies designers' ideas. Second, it is important advantage that the user can check side-effects at the very early stage of design. And finally, representing a set of components performing a function as a physical feature is useful scheme for understanding and reusing design results. On the other hand, the following issues are clarified: First, it was difficult for designers to symbolize functions because functions are ambiguous in nature and they were not educated to do so. In spite of this difficulty, they felt that symbolization of functions would be useful for organizing and re-using their knowledge. And second, since even in early stage of design, designers calculate critical quantitative values, the FBS Modeler will become more industrial relevant when it can connect quantitative modelers easily.

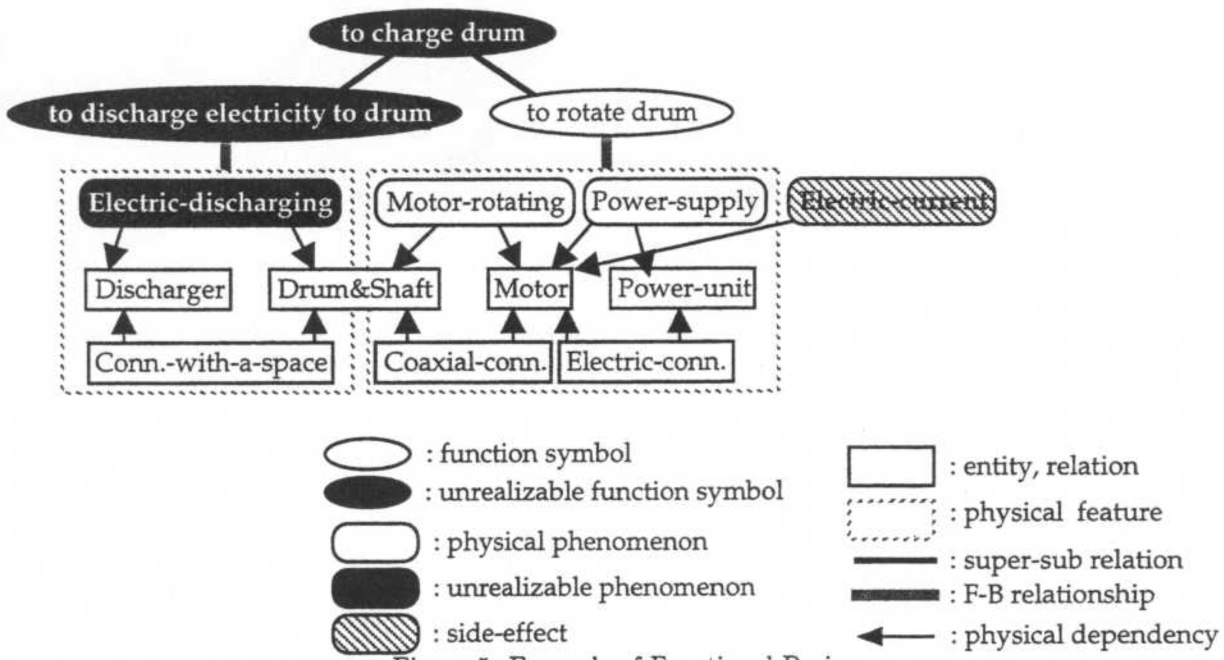

Figure 5: Example of Functional Design

4 Related Work

Recently, research interests in functional reasoning are increasing (e.g., [8, 13]). The FBS modeling can be compared with related work from the following viewpoints.

1. Applications

   We view that the difference between analytical tasks (*e.g.* diagnosis [14, 15], design verification [6], and tutoring [16]) and synthetic tasks (*e.g.* design [3, 17]) is critical. While an analytical task is a process that transforms structural descriptions into functional descriptions understandable for the user, a synthetic task is a process that transforms functional descriptions representing the user's intention into structural descriptions. This paper discussed indispensable features of functional modeling for supporting synthetic tasks.

2. Representational grounding

   Functions can be defined with regard to concepts such as behavior and structure. Most of existing work deals with behavioral function and they can be classified further as follows:

   (a) Device-oriented representation

   Many researchers (*e.g.*, [14, 7, 15, 3]) represent structure as a network of components that have flow of energy, material, and information. This flow then defines behavior. As a result, functions are described in a device-oriented way.

   (b) Process-oriented representation

   Forbus [16] represented behavioral functions based on QPT. Faltings [18] proposed place vocabulary that describes relative motions of kinematic pairs. Function in these approaches are related to physical phenomena occurring among some components rather than components. Our approach presented in this paper also follows this category.

   We view that the process-oriented representation is more appropriate for mechanical design for the following reasons:

   - The device-oriented representation is appropriate to the domains such as electrical circuits in which each function corresponds well to each component. However, since, as Faltings pointed out [18], a function in mechanical domain corresponds to an interaction among some components rather than a component, the process-oriented representation is more appropriate. For example, the function of linear guide is hardly represented by the device-oriented representation.
   - The device-oriented representation is appropriate to the domains in which the number of elemental components is limited. However, in the mechanical domain, while the number of components is unlimited, the number of physical phenomena seems to be much smaller than that of components. This is the reason why we use the *physical features* as building blocks of behavior for the FBS Modeler.

3. Formalization of functional representation

   Concerning formalization of functional representation, Keuneke [5] proposed four function types; namely, ToMake, ToMaintain, ToPrevent, and ToControl. Iwasaki [6] represented behavioral functions based on causality among state transitions.

Welch [3] proposed bond graph based functional representation.

Our formalization is less formal than these in that a symbol of function is represented just as "to verb objects." However, this makes it easier to describe functions more flexibly and, then, physical semantics of the symbol can be added flexibly by using the F-B relationships when the description becomes precise enough. These features are indispensable for supporting conceptual design.

4. Function redundancy

   Our approach agrees with Keuneke's approach [5] in that functional structure does not always correspond to physical structure, while some others (*e.g.* [2, 15]) assume so. We further demonstrated the importance of this with the functionally redundant in Section 3.3. Function redundancy is an opposite operation of the function sharing [19] to obtain higher reliability.

5 Conclusions

In this paper, we have proposed FBS modeling to represent functions in the context of design and the FBS modeler to assist functional design. We have demonstrated practical advantages of the FBS modeling with an innovative design methodology of design for function redundancy. The main feature of the FBS Modeler is that it is developed for supporting conceptual design.

Assisting designers in conceptual design requires that the system should support the function definition process without reference to behavior and structure of the artifact and the function embodiment process to determine behavior and structure. In the FBS Modeler, the behavioral knowledge based on *physical features* and the subsystem QPAS allow the designer to find out behaviors that realize required functions.

References

[1] W. Rodenacker. *Methodisches Konstruieren.* Springer-Verlag, Berlin, 1971.

[2] G. Pahl and W. Beitz. *Engineering Design: A Systematic Approach.* Springer-Verlag, Berlin, 1988.

[3] R. V. Welch and J. R. Dixon. Representing function, behavior and structure during conceptual design. In D. L. Taylor and L. A. Stauffer, editors, *Design Theory and Methodology — DTM'92—*, pp. 11-18. ASME, 1992.

[4] L. D. Miles. *Techniques of Value Analysis and Engineering.* McGraw-Hill, New York, 1972.

[5] A. M. Keuneke. Device representation: The significance of functional knowledge. *IEEE Expert*, Vol. 6, No. 2, pp. 22–25, 1991.

[6] Y. Iwasaki, R. Fikes, M. Vescovi, and B. Chandrasekaran. How things are intended to work: Capturing functional knowledge in device design. In *Proceedings of IJCAI'93*, pp. 1516 -1522, 1993.

[7] M. Pegah and W. E. Bond et al. Functional rpresenting and reasoning about the F/A-18 aircraft fuel system. *IEEE Expert*, Vol. 4, pp. 65–71, 1993.

[8] A. N. Kumar, editor. *AAAI-93 Workshop Working Notes of Reasoning About Function.* AAAI, 1993.

[9] K. Forbus. Qualitative process theory. *Artificial Intelligence*, Vol. 24, No. 3, pp. 85–168, 1984.

[10] T. Tomiyama, T. Kiriyama, and H. Yoshikawa. Conceptual design of mechanisms: A qualitative physics approach. In A. Kusiak, editor, *Concurrent Engineering: Automation, Tools, and Techniques*, pp. 131–152. John Wiley, 1992.

[11] M. Ishii, T. Tomiyama, and H. Yoshikawa. A synthetic reasoning method for conceptual design. In *IFIP World Class Manufacturing '93*, Amsterdam, 1993. North-Holland.

[12] Y. Umeda, T. Tomiyama, H. Yoshikawa, and Y. Shimomura. Using functional maintenance to improve fault toleneance. *IEEE EXPERT*, Vol. 2, No. 3, pp. 25 -31, June 1994.

[13] J. Hodges, editor. *AAAI-93 Workshop Workshop Notes of Representing and Reasoning About Device Function.* AAAI, 1994.

[14] A. Abu-Hanna, R. Benjamins, and W. Jansweijer. Device understanding and modeling for diagnosis. *IEEE Expert*, Vol. 6, No. 2, pp. 26 - 32, 1991.

[15] J. A. Bradshaw and R. M. Young. Evaluating design using knowledge of purpose and knowledge of structure. *IEEE Expert*, Vol. 6, No. 2, pp. 33–40, 1991.

[16] K. D. Forbus. Qualitative reasoning about function: A progress report. In A. N. Kumar, editor, *AAAI-93 Workshop Working Notes of Reasoning About Function*, pp. 31 - 36. AAAI, 1993.

[17] D. W. Franke. Deriving and using descriptions of purpose. *IEEE Expert*, Vol. 6, No. 2, pp. 41–47, 1991.

[18] B. Faltings and K. Sun. Causal inversion: applying kinematic principles to mechanism design. In Y. Ishida *et al.*, editor, *IJCAI 93 Workshop: Engineering Problems for Qualitative Reasoning*, pp. 81 - 86. IJCAI, 1993.

[19] K. T. Ulrich and W. P. Seering. Function sharing in mechanical design. In *Proceedings of AAAI-88*, pp. 342–346, 1988.