Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1996), *10*, 275–288. Printed in the USA.

Supporting conceptual design based on the function–behavior–state modeler

YASUSHI UMEDA,[1] MASAKI ISHII,[1] MASAHARU YOSHIOKA,[1] YOSHIKI SHIMOMURA,[2] AND TETSUO TOMIYAMA[1]

[1] Department of Precision Machinery Engineering, Graduate School of Engineering, the University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan
[2] Mita Industrial Co., Ltd., Tamatsukuri 1-2-28, Chuo-ku, Osaka, Japan

(RECEIVED April 10, 1995; ACCEPTED December 15, 1995)

Abstract

The relative significance of conceptual design to basic design or detail design is widely recognized, due to its influential roles in determining the product's fundamental features and development costs. Although there are some general methodologies dealing with functions in design, virtually no commercial CAD systems can support functional design, in particular so-called synthetic phase of design. Supporting the synthetic phase of conceptual design is one of the crucial issues of CAD systems with function modeling capabilities. In this paper, we propose a computer tool called a Function–Behavior–State (FBS) Modeler to support functional design not only in the analytical phase but also in the synthetic phase. To do so, the functional decomposition knowledge and physical features in the knowledge base of the modeler, and a subsystem Qualitative Process Abduction System (QPAS) play crucial roles. Modeling scheme of function in relation with behavior and structure and design process for conceptual design in the FBS Modeler are described. The advantages of the FBS Modeler are demonstrated by presenting two examples; namely, an experiment in which designers used this tool and the design of functionally redundant machines, which is a new design methodology for highly reliable machines, as its application.

Keywords: Conceptual Design; FBS Modeler; Function Modeling; Functional Reasoning; Function Redundancy

1. INTRODUCTION

Let us consider a design process that consists of functional design (or conceptual design), basic design, and detail design. Because a designer should determine the functional structure of a design object and basic physical mechanisms that realize the functional structure in functional design, functional design is more important than basic design or detail design (Yoshikawa & Gossard, 1989). However, traditional CAD technology, dealing mainly with geometry, cannot treat function well, because function is a more abstract concept than the geometric information that can be generated only after functional design. Therefore, representation and manipulation of function are crucial issues for constructing a CAD system that supports conceptual design. Recent need for high-performance and innovative design require development of such a CAD system.

Although there are some general methodologies dealing with functions (e.g., Rodenacker, 1971; Pahl & Beitz, 1988), virtually no commercial CAD system can support functional design, in particular, the so-called synthetic phase of design. In this paper, we propose a computer tool called a Function–Behavior–State (FBS) Modeler to support functional design not only in the analytical phase but also in the synthetic phase. We clarify advantages of the FBS Modeler by presenting two examples; namely, an experiment in which designers used this tool and the design of functionally redundant machines. Function redundant design is a new design methodology for highly reliable machines.

Reprint requests to: Dr. Yasushi Umeda, Department of Precision Machinery Engineering, 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; E-mail: {umeda, tomiyama, ishii, yoshioka}@zzz.pe.u-tokyo.ac.jp.

The rest of the paper is organized as follows: Section 2 analyzes existing methods to represent functions and presents a theoretical background of the FBS Modeler. Section 3 describes the FBS Modeler and its usage in functional design. Section 4 illustrates an experimental use of the FBS Modeler. In Section 5, we propose a design methodology of functionally redundant machines as an application of the modeler. We discuss advantages and issues of the modeler in Section 6, and Section 7 concludes this paper.

2. FUNCTION–BEHAVIOR–STATE MODELING

In this section, we discuss problems in representing function in the conventional approaches and propose a new framework called a FBS diagram.

2.1. Representation of function

There is no clear and uniform definition of *function* because function is an intuitive concept depending on the designer's intention.

Rodenacker (1971) defines function as a relationship between input and output of energy, material, and information. This definition is accepted widely in design research (e.g., Pahl & Beitz, 1988; Welch & Dixon, 1992). However, this representation has limitations; for example, the function of a fixture "to fix something for manufacturing it" and the function of a linear guide "to guide the motion of an object on a straight line" cannot be represented. Value engineering (VE) represents a function in the form of *to do something* (Miles, 1972). Although this definition is general, due to the lack of clear description about relationships between function and structure, this representation is not powerful enough for design applications.

We define a *function* as "a description of behavior abstracted by human through recognition of the behavior in order to utilize the behavior" (Umeda et al., 1990). In other words, it is difficult to distinguish clearly function from behavior and it is not meaningful to represent function independently of the behavior from which it is abstracted. Here we represent a function as an association of two concepts; that is, its symbol represented in the form of *to do something*, as VE proposed, and a set of behaviors that can exhibit the function. 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 relationships between functions and behaviors are subjective and many-to-many correspondent; for example, we might say that some behaviors such as "hitting a bell" and "oscillating a string" realize a function "to make a sound." We call these relationships *F–B relationships*.

In this framework, we assume that the representation of function includes human intention as the symbols and F–B relationships, whereas the representation of behavior of an entity can be determined more objectively based on physical principles. Here, a state of an entity is defined by a set of attributes and relations among relevant entities.[1] Figure 1 shows a part of the state description of a paper weight. In Figure 1, "Paper Weight" and "Paper" are entities and they have the relation "on," which means that the paper weight is on the paper. "Paper Weight" has some attributes that have values; namely, "Weight: 1 kg," "Volume: 100 cm^3," and "Density: 10 g/cm^3." These attributes are also related with each other.

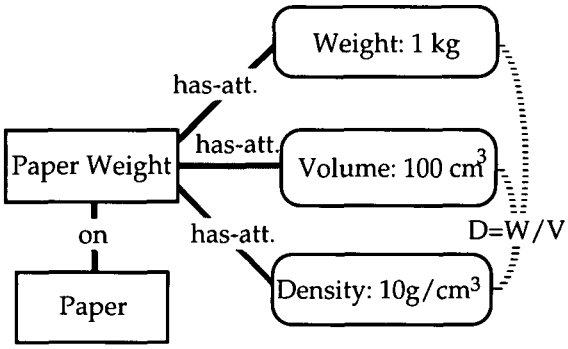

Fig. 1. State of paper weight (Umeda et al., 1990).

Introducing a discrete unit time, we define *behavior* as "sequential state transitions along time," and assume that *physical phenomena* determine behavior of an entity. In other words, we can reason out all possible behaviors of an entity from an initial state with a set of physical phenomena. We call these relationships between behaviors and states *B–S relationships*.

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 with resistance, voltage, current, and so on, it can also be captured as motion of electrons. To represent this difference, we introduce *aspects*. An aspect is a collection of all relevant entities, attributes, relations, and physical phenomena of the current interest. Figure 2 illustrates the relationships among function, behavior, and state.

2.2. Functional decomposition

2.2.1. *Function–behavior–state diagram*

In design, a designer decomposes the required functions into subfunctions hierarchically until arriving at substantial components (e.g., Pahl & Beitz, 1988). Therefore, it is essential for supporting design to represent a design object hierarchically in such a way that its representation becomes gradually concrete over hierarchy. We assume that such hi-

[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, the structure of a machine might change when a fault occurs.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

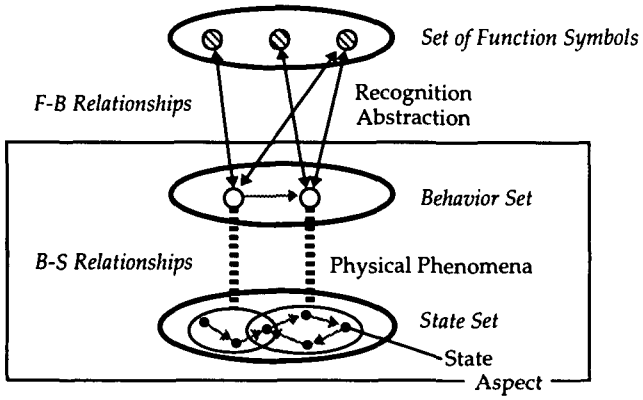

Fig. 2. Relationship among function, behavior, and state.

erarchies can only be constructed subjectively from the viewpoint of function rather than objectively or physically.

We represent a design object hierarchically as shown in Figure 3. We call this scheme a *FBS diagram*. This figure indicates that the designer should concurrently describe symbols of functions and their actualizing behaviors with appropriate aspects in a hierarchy. Because aspects are physically related with each other, consistency among aspects and, therefore, consistency of behaviors and states in each aspect can be maintained by a computer with a sufficient amount of aspects and physical knowledge in these aspects. For example, we have proposed the *metamodel* mechanism that manages relationships among aspects (Tomiyama et al., 1992).

The main ideas of the FBS diagram are:

- to distinguish subjective parts of a design object (*function symbols* and F–B *relationships*) and objective parts (*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; in other words, functions relate subjective concepts and objective concepts; and
- to represent a design object hierarchically to support a modeling process that details functional and behavioral descriptions concurrently.

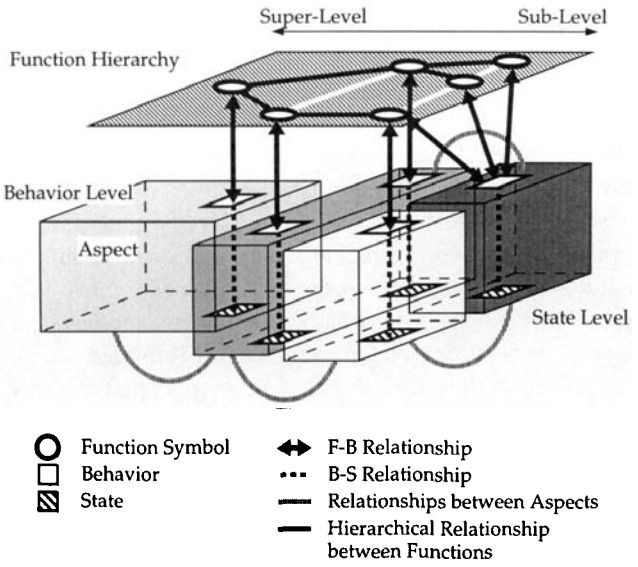

Fig. 3. Function–behavior–state diagram.

Because of 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 in the objective parts, and propose modification for the inconsistencies.

2.2.2. *Task decomposition and causal decomposition*

Functional decomposition is one of the key issues in the synthetic phase of design. Let us examine this functional decomposition in detail.

Based on results of experiments described in Section 4, we propose that decomposition of functions can be classified into the following two categories. Here, we assume that a function f is decomposed into subfunctions $f_1, f_2, \ldots, f_n$ that are embodied by behaviors $b_1, b_2, \ldots, b_n$.

Causal decomposition: For instance a function f "to generate light" can be decomposed as follows:

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

In this case, b_2 is indispensable for activating b_1. We call this kind of decomposition *causal decomposition*, for which a definition is that behaviors resulting from this decomposition are causally related with each other.

Task decomposition: For example, a function f "to make salt" can be decomposed as follows:

f_1: to collect sea water	b_1: water pouring
f_2: to boil salty water	b_2: water boiling.

This kind of decomposition is *task decomposition* rather than causal decomposition. Namely, b_1 and b_2 are not causally related with each other, because they can occur independently.[2]

While the task decompositions are not physically derivable, the causal decompositions can be reasoned out by searching causal relationships through knowledge about behavior. Therefore, in the implementation of the FBS Modeler described in Section 3, while the task decompositions are described as a part of functional knowledge explicitly

[2] Of course, they should be related structurally to perform the target function f.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

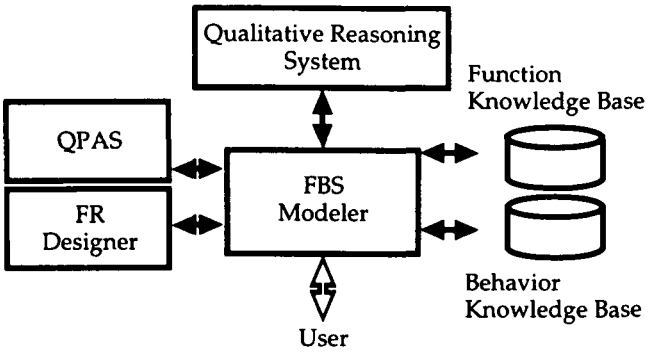

Fig. 4. Architecture of the FBS modeler.

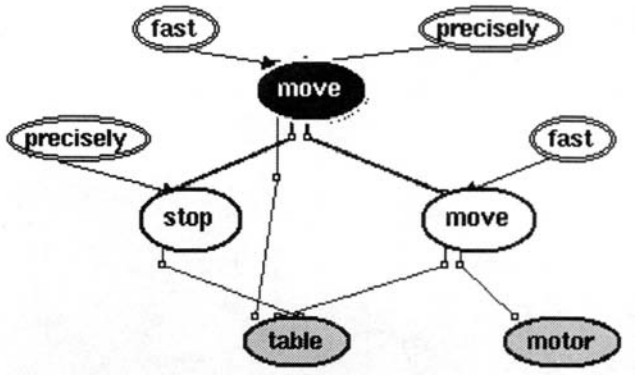

Fig. 5. Example of functional decomposition knowledge.

and executed manually, the causal decompositions are executed by the system with physical knowledge. The latter helps the designer to find out new design solutions. By applying the task decompositions hierarchically, the designer constructs a task-decomposed functional hierarchy from the required functions. In contrast, for the causal decompositions, a subsystem called Qualitative Process Abduction System (QPAS) (Ishii et al., 1993) helps the designer to find out appropriate mechanisms for realizing task-decomposed functions with physical knowledge.

3. FUNCTION–BEHAVIOR–STATE MODELER

We have developed a computer tool called a *FBS Modeler* for supporting functional design based on the FBS diagram. Because of implementational limitations, the modeler can handle only one *aspect* at one time that represents behaviors at the bottom level in Figure 3.

Figure 4 shows the architecture of the FBS Modeler. The qualitative reasoning system (Tomiyama et al., 1992), based on Qualitative Process Theory (Forbus, 1984), gives a representational scheme of behaviors and states and performs behavioral simulation. QPAS (Ishii et al., 1993) that derives physical phenomena from the given state transitions is incorporated for supporting the causal decomposition. The Function Redundancy (FR) Designer will be described in Section 5. Knowledge about function is described in the function knowledge base in the form of function prototypes described in Section 3.1.1. The behavior knowledge base consists of physical features, physical phenomena, relations, and entities described in Section 3.1.2.

Table 1. *Definition of function prototype*

Item	Contents
Name	*verb + objects + modifiers*
Decomposition	networks of subfunctions
F–B relationship	physical features

3.1. Knowledge representation

3.1.1. Functional knowledge

To construct a knowledge base of functions, we collect prototypes of functions from existing design results. Table 1 shows the scheme of the function prototype.

Name is the symbol for representing a designer's intention in the form of "to *verb objects modifiers*." Here, *objects* represent entities and things related to the function such as "gear" and "electricity." *Modifiers* qualify a function and include such terms as "fast" and "reliably."

Decomposition describes feasible candidates for decomposing the function in the form of networks of subfunctions. This hierarchy is composed of either abstract–concrete relations or whole–part relations. Here, only task decompositions are included in the decomposition knowledge. Figure 5 shows an example of the decomposition knowledge. This network denotes that the target function "to move a table fast and precisely"[3] (shown as a black node) can be decomposed into two subfunctions; namely, "to move the table with a motor fast" and "to stop the table precisely."

The *F–B Relationship* describes candidates of embodiments of the function in the form of *physical features* described below.

3.1.2. Physical feature

Generally speaking, a designer considers components and their behaviors at the same time. Moreover, the F–B relationship implies that it is useful to collect building blocks of behaviors that correspond to functions. For representing such knowledge, we have proposed *physical features* that are building blocks consisting of components and physical phenomena occurring on the components (Tomiyama et al., 1992). We introduce the physical feature to relate functions to behaviors and states. Our representation of behaviors and

[3] In Figure 5, gray nodes and double oval nodes represent objects and modifiers of the function name, respectively; namely, the function "to move a table fast and precisely" consists of the verb node "move," the object node "table," and the modifier nodes "fast" and "precisely."

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

Table 2. *Definition of physical phenomenon (Tomiyama et al., 1992)*

Item	Contents
Phenomenon	name
Supers	super classes
Conditions	
prerequisites	needed entities
references	needed causal dependencies among prerequisites
relations	needed relations
q-conditions	parametric conditions
s-conditions	parametric conditions given outside of QPT
Influences	
quantities	definition of parameters
q-relations	proportional equations
influences	differential equations

Table 3. *Definition of electrical discharging*

Item	Contents
Phenomenon	ElectricalDischarging
Supers	Fundamental
Conditions	
prerequisites	discharger = Discharger, device = ElectricalNonConductor
references	
relations	connectionsWithASpace(discharger,device)
q-conditions	(discharger voltage) > zero
s-conditions	(discharger sw) = on
Influences	
quantities	(device charge) = (− zero +)
q-relations	(device charge) direct (discharger voltage)
influences	

states of design objects is based on Qualitative Process Theory (QPT) (Forbus, 1984).

In the FBS Modeler, a physical feature is represented as a network of three kinds of elements (we call them *behavior nodes*); namely, entities, relations, and physical phenomena. An *entity* is a component such as a gear, a spring, and a shaft and plays the same role as an individual in QPT. *Relations* represent structural relationships among entities such as "on," "above," and "connected." A *physical phenomenon* is the same concept as a process in QPT and defined in Table 2. If a phenomenon is activated by satisfying its conditions, it adds parameters and qualitative equations defined in the influences in Table 2.

Each physical feature is constructed by the designer so as to be a meaningful block for representing a function. For example, Figure 6 shows an example of a physical feature in which a phenomenon "ElectricalDischarging," defined in Table 3, occurs between a discharger and a nonconductor. In this figure, arcs between nodes represent physical dependencies; that is, this phenomenon depends on the discharger and the nonconductor, which are related by the relation "ConnectionWithASpace," as specified in the conditions in Table 3.

We view that it is difficult to define general *primitives* for representing functions and behaviors, while it might be possible in some limited domains. Instead, our approach to represent functions and behaviors is to collect various function prototypes and physical features from existing designs and to construct a large knowledge base of them. We believe that the system will help designers to create new design solutions by searching through the knowledge base.

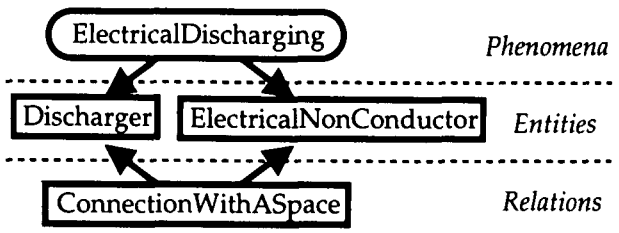

Fig. 6. Example of a physical feature.

3.2. Functional design on the FBS modeler

Here, we consider that functional design is to construct a consistent and feasible FBS model of a design object by detailing and embodying the required functions on the FBS Modeler. Figure 7 illustrates the basic flow of the functional design on the FBS Modeler.

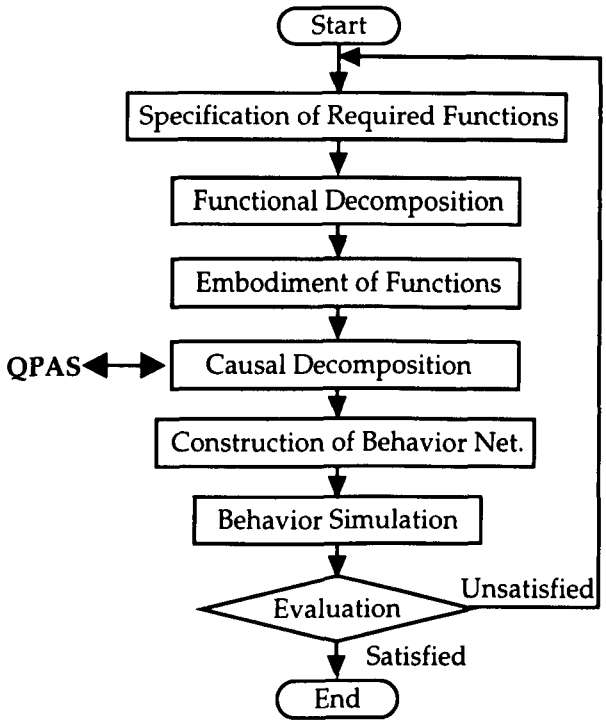

Fig. 7. Basic flow of functional design.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

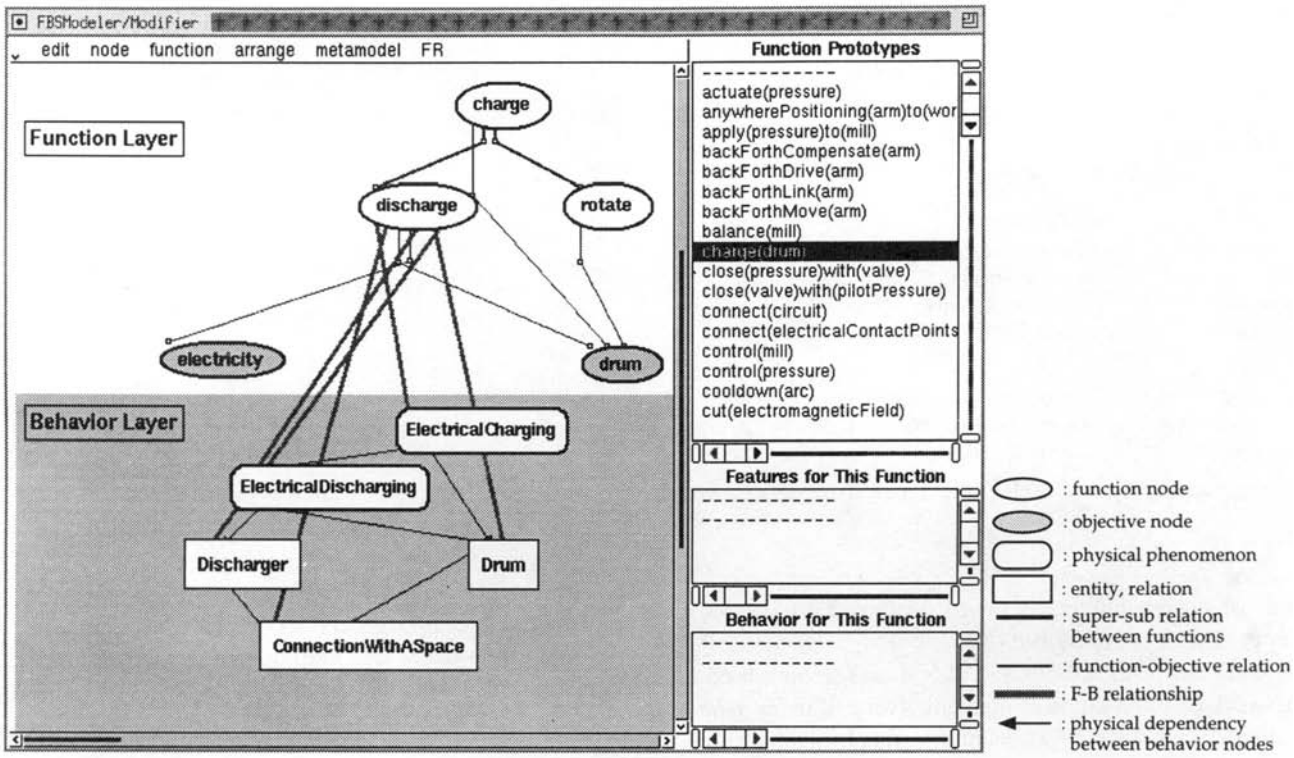

Fig. 8. Example of functional design (1).

Specification of required functions. The designer inputs required functions by choosing function prototypes.

Functional decomposition. The designer recursively decomposes the required functions into subfunctions by applying the decomposition knowledge of function prototypes and, as a result, constructs a functional hierarchy. These decompositions are task decomposition. By collecting a large amount of decomposition knowledge, the system can help the designer to find out new solutions of functional decomposition. This is one feature of the modeler for supporting the synthetic phase of design.

Embodiment of functions. Next, the designer instantiates physical features that can embody the hierarchy by using the F–B relationships of prototypes for each function. Again, by collecting a large amount of knowledge about the F–B relationships, the system can help the designer to find out new solutions by combining them. This is the second feature of the modeler to support the synthetic phase.

For example, Figure 8 depicts that a required function "to charge drum," which often appears in design of photocopiers, is task-decomposed into two subfunctions "to discharge electricity to drum" and "to rotate drum" and the subfunction "to discharge electricity to drum" is embodied by a physical feature represented as a network of five behavior nodes in the *Behavior Layer*.[4]

[4] The subfunction "to rotate drum" is not embodied yet in this figure.

Causal decomposition. After instantiating physical features, the designer often finds out that some of them cannot occur. For example, the physical feature shown in Figure 8 is not physically adequate, because the condition of the phenomenon "ElectricalDischarging," defined in Table 3, is not satisfied yet. In such a case, QPAS (Ishii et al., 1993) reasons out candidates of additional physical features to satisfy the physical conditions.

After the designer specifies a physical feature *pf*, the system helps the designer to find out additional features *pf'* as follows.

a. *Candidate generation:* To realize the feature *pf*, all phenomena in *pf* should occur. If a physical phenomenon is inadequate to occur, it may have the following two kinds of incompleteness:

   Prerequisites: To activate a phenomenon, all prerequisites of its definition should be connected to it. If a phenomenon has unconnected prerequisites, these prerequisites should be added to the model and connected to it. In this case, QPAS searches for physical features that include the unconnected prerequisites.

   Parametric conditions: To activate a phenomenon, all parametric conditions (viz., q-*conditions* and s-*conditions*) of its definition should be satisfied. The designer should decide these conditions to be

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

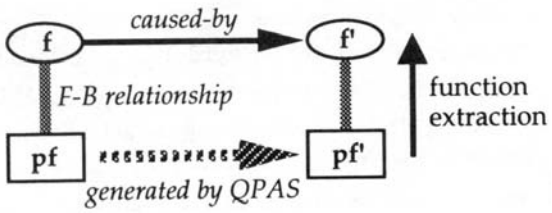

Fig. 9. Flow of the causal decomposition.

satisfied either by giving it as initial conditions or by adding physical features that can realize the conditions. While the designer should describe the initial conditions to the model in the former case, in the latter case, by providing QPAS with each parametric condition, the system reasons out candidates of physical features that include phenomena that can change the value of the parameter in the condition; in other words, the phenomena that have the same parameter in their q-*relations* or *influences*.

b. *Feature instantiation:* After the designer selects a physical feature among the derived candidates, the feature is instantiated in the model and connected to *pf* by using physical dependencies.

c. *Function extraction:* Next, a function is added for explaining the additional feature *pf'*. The FBS Modeler reasons out function prototypes that have *pf'* in the F–B relationships. After the designer selects a prototype, it is instantiated and connected to *pf'* by the F–B relationship. Moreover, the target function *f* is connected to the instantiated function *f'* by a *caused-by* relationship that denotes that *f* is caused by *f'* (see Fig. 9).

d. *Repetition:* By repeating this cycle, a network of behavior nodes is constructed so as to realize the target physical feature *pf*.

In this way, the modeler supports the designer to find out new solutions by reasoning out additional physical features.

Figure 10 shows an example of adding a physical feature to the example shown in Figure 8. In Figure 8, the following two conditions of the phenomenon "ElectricalDischarging" have not been satisfied yet (see Table 3); "(discharger voltage) > zero," which means voltage of the discharger should be higher than zero, and "(discharger sw) = on,"

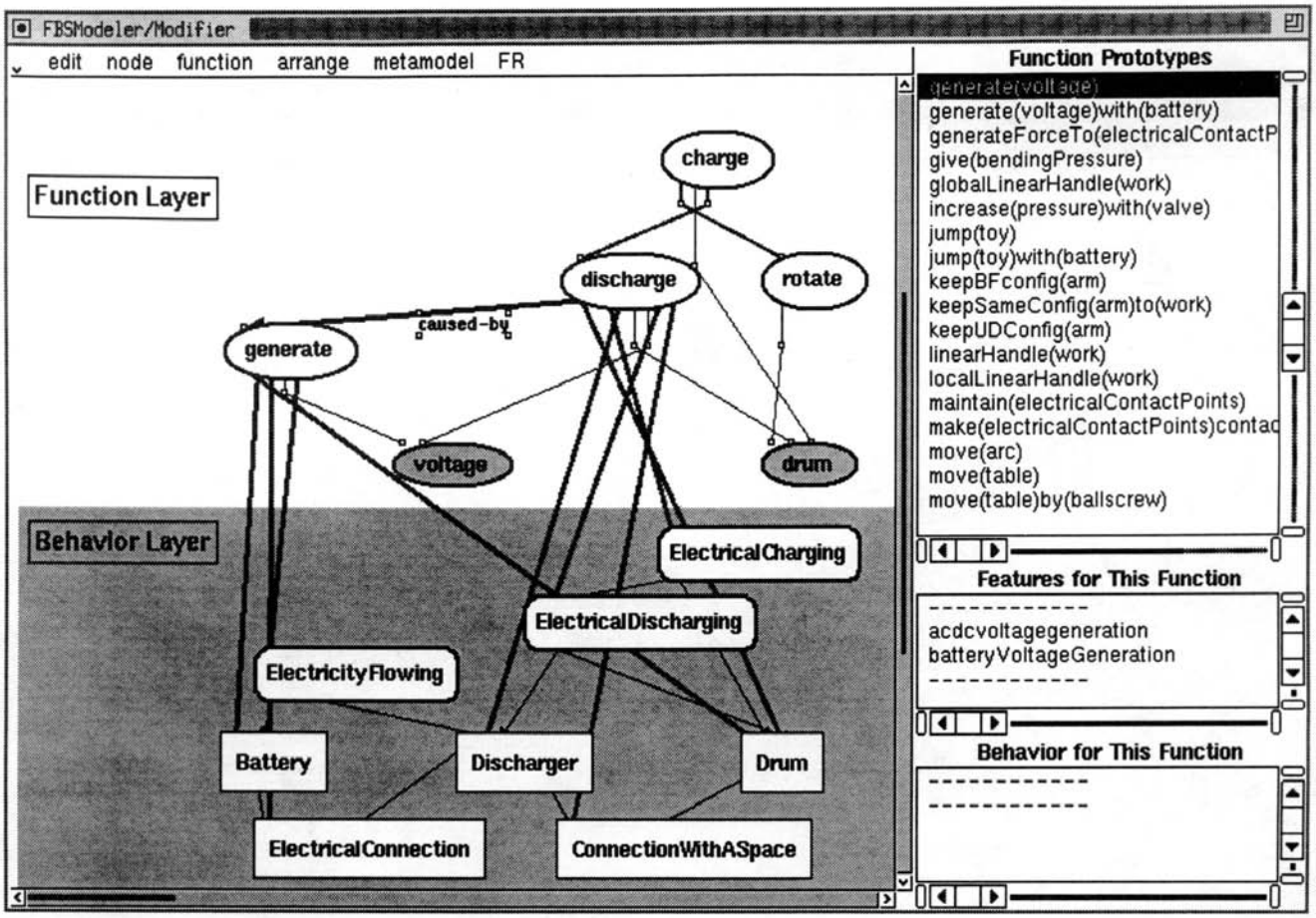

Fig. 10. Example of functional design (2).

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

which means that the discharger should be switched on. In this case, while "(discharger sw) = on" was given as an initial condition by the designer, "(discharger voltage) > zero" was provided to QPAS. By selecting a candidate derived by QPAS, the designer added a feature in which a phenomenon "ElectricityFlowing" occurs on a battery. Moreover, by selecting an extracted function, the system added the function "to generate voltage" and connected it to the instantiated feature. The function "to discharge electricity to a drum" is related to this function by the *caused–by* relation.

Construction of the behavior network. Next, the designer should construct a physically consistent network of behavior nodes (called a *behavioral network*) by connecting instantiated physical features so as to complete the functional hierarchy. This is done by unifying the same entities in different features.

Behavior simulation and evaluation. After constructing the behavior network, the qualitative reasoning system executes behavior simulation on the behavior network. The reasoning system compares the initial FBS model with the result of simulation and indicates the following information. This evaluation supports the designer in the analytical phase of design at an early stage of design.

Unrealizable phenomena: If physical phenomena designated by the designer do not occur in the simulated network, some conditions should be inadequate.

Side-effects: Phenomena that are not expected to occur may cause side-effects that the designer did not notice.

Unrealizable functions: If functions have unrealizable phenomena in their F–B relationships or unrealizable subfunctions, they will not be realized.

Figure 11 shows the FBS model of Figure 10 after this evaluation. The black and hatched nodes indicate unrealizable phenomena and side-effects, respectively, and the black oval nodes represent unrealizable functions. From this figure, the designer understands that the required function "to charge the drum" is not realized when the drum is put in the light because the drum is photo-semiconductor, and finds out that he/she should enclose the drum.

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

4. EXPERIMENTAL USE OF THE FBS MODELER

To evaluate the performance of the FBS Modeler, we asked three groups of designers to use the modeler. They needed about a week to learn the modeler.

A group from a construction company used the modeler for conceptual design of houses. In this experiment, the designers determined the configurations of living rooms, kitchens, bathrooms, and so on by using this modeler. The modeler

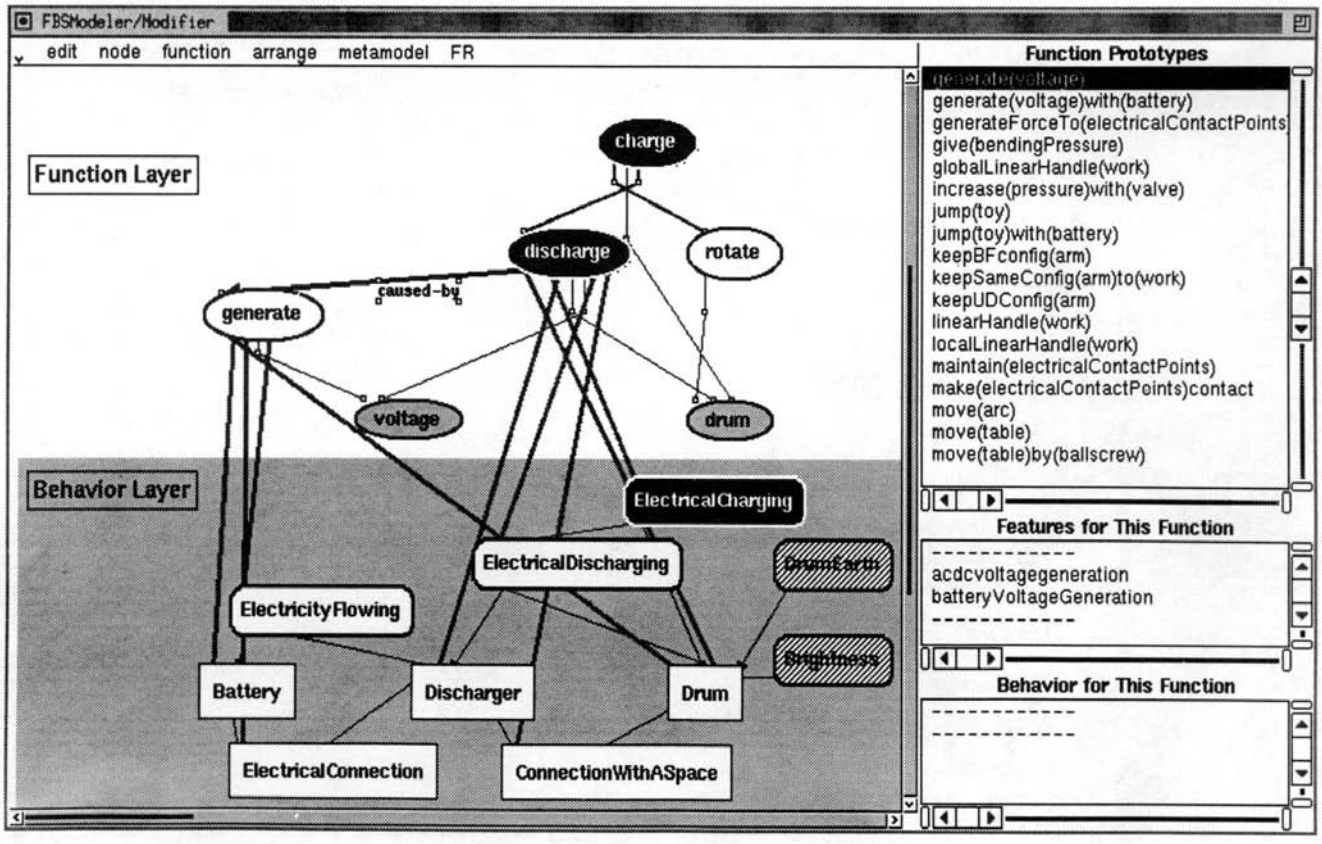

Fig. 11. Example of evaluation.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

helped them to collect and describe functional and behavioral knowledge about rooms and how to configure the rooms by executing the behavior simulation about air flow, human movement, noise propagation, and so on.

Other groups from an electronics company represented their products (a robot arm [see Fig. 12] and a videotape recorder). In these experiments, the designers focused on collecting functional knowledge and describing basic structure of the products rather than designing new products. The knowledge representational scheme of the modeler helped them to describe tacit functions explicitly. The simulation helped to verify correctness and consistency of described knowledge.

As a result of this experimental use, the following advantages are identified.

1. The FBS Modeler is useful for functional design not only in the domain of mechanical design but also house design. This advantage is due to the modeler's capability to represent abstract concepts, which could not be represented with traditional CAD systems in the form of functions and qualitative physics. They concluded that such knowledge is useful not only for novice designers but also for experts, because it clarifies designer's idea.

2. They felt that if enough knowledge and FBS models of existing products were provided, they could design new products by referring to models of similar products in the FBS Modeler.

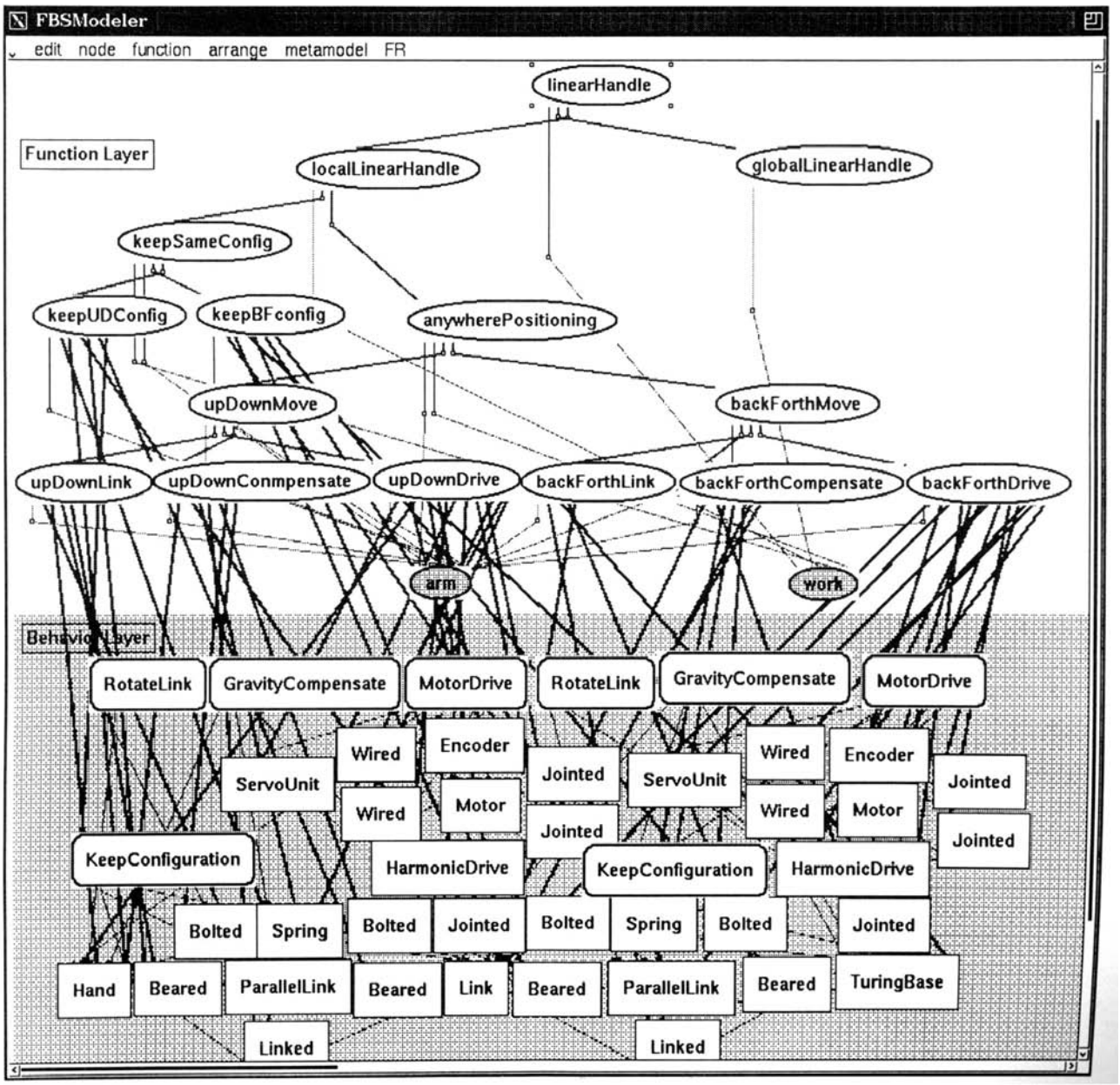

Fig. 12. The FBS model of a robot arm.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

3. It is important that the user can check side-effects at the very early stage of design. For example, they checked air flow to avoid spreading foul odors from the toilet in the design of house.
4. They considered that representing a group of components performing a function as a physical feature is useful for understanding and reusing design results.

However, the following problems were pointed out.

1. It was difficult to symbolize functions, because functions were ambiguous in nature and the designers were not educated to do so. In spite of this difficulty, they felt that symbolization of functions is useful for organizing and reusing their knowledge.
2. Because even in the early stage of design designers calculate critical quantitative values, the FBS Modeler would become more industrially relevant, if it could introduce quantitative modelers easily. To solve this problem, we are developing a framework called a *Knowledge Intensive Engineering Framework* (Tomiyama et al., 1994) that integrates various kinds of modelers including geometric modelers, FEM (Finit Element Method) modelers, and, of course, the FBS Modeler.
3. They wanted to evaluate functions more quantitatively to select better solutions from alternatives. We view that the *modifier* shown in Table 1 is the key for such evaluation. For this purpose, we are developing a quantitative evaluation method called *Amount of Function*, which evaluates the FBS models based on the modifiers (Shimomura et al., 1995).

5. EXAMPLE OF APPLICATION: DESIGN FOR FUNCTION REDUNDANCY

In this section, we illustrate a design methodology for highly reliable machines as an application of the functional design to demonstrate the advantage of the FBS Modeler.

5.1. Function redundancy

Principal strategies of the traditional reliability design methodologies (e.g., Ireson, 1966) are to make each component more reliable and to add redundancy to the machine. While there is a certain limit to the former, the latter *part redundancy* often results in undesirable cost, weight, and complexity of the machine due to additional redundant parts.

We proposed a new idea called *function redundancy* (Umeda et al., 1992). Function redundancy is achieved by using potential functions of existing parts in a slightly different way from the original design. For example, in case of emergency when the engine stops, a car with a manual transmission can run for a while with its starting motor. This *function redundancy* depends on the fact that the starting motor performs its potential function "to generate driving force" instead of the original function "to start the engine" by changing the power flow of the car. For achieving similar redundancy under the part redundancy strategy, we need to add another engine to the car.

5.2. Design for function redundancy

Here, we propose design methodology for function redundancy based on the FBS Modeler. The FBS Modeler is useful, because:

1. The FBS Modeler represents functions and many-to-many correspondent F–B relationships.
2. The FBS Modeler can reason about potential functions by searching through the function and behavior knowledge bases.

In the FBS Modeler, the function redundancy is modeled as shown in Figure 13; the target function F_a is decomposed into subfunctions F_b and F_c, which are embodied by Behavior nodes (BNodes) 1, 2, 3, and 4 and 5, 6, 7, and 8, respectively, in the normal state. If, for example BNode 5 is lost by a fault, the subfunction F_c and, therefore, the target function F_a are lost. Here, if the prototype of the lost function F_c includes other physical features that can be activated in the faulty machine, the lost function can be recovered. This is the function redundancy. In the example in Figure 13, because the prototype of the lost function F_c has another physical feature that consists of BNodes 1, 2, 9, and 10 in its F–B relationship, the *potential* function F_c' appears by activating additional phenomena BNodes 9 and 10 and, therefore, the target function F_a are recovered.

We have developed a subsystem for function redundancy, called *Function Redundancy* (FR) *Designer* (see Fig. 4). The FR Designer searches for candidates of function redundancy by the above-mentioned method and eval-

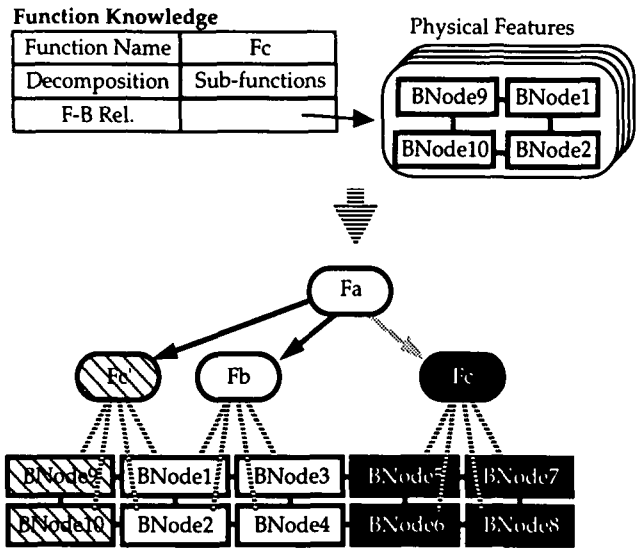

Fig. 13. Function redundancy in the FBS modeler.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

uates them. The input to this system is an FBS model of a design object and the output is an FBS model that has function redundancies. By inputting the FBS model, the design for function redundancy is executed as follows:

1. *Select a target function.* The designer selects a target function to which function redundancy is added.
2. *Derive candidates of function redundancy.* The system generates candidates of function redundancy for the target function by searching for potential functions in the initial FBS model as described above.
3. *Modify design object for function redundancy.* After selecting a candidate, the designer should modify the design object for activating the required behaviors for the function redundancy. This modification can be realized in many ways; namely, adding parts, changing existing parts, changing control sequence programs, and so on. It is also necessary to add some switching mechanisms for activating potential functions.
4. *Evaluate result.* The FR Designer indicates the following information for supporting the designer to evaluate the design.

   Robustness: The system shows ratio of redundant functions achieved by the FR candidate by comparing the original functional hierarchy and the modified one. Here, we assume that redundancy in the functional level increases the reliability of the design object.

   Redundancy: The system shows ratio of added behavior nodes for realizing the function redundancy. Namely, redundancy in behavior level implies additional costs or difficulty for realizing the selected FR candidate.

Therefore, the designer should modify the design object to get most Robustness and least Redundancy.

5.3. Example

As an example, we designed a functionally redundant photocopier. Figure 14 depicts the original structure of the copier. The target function is a function "to charge the drum," which is performed by the main charger.

Figure 15 depicts the result of the function redundant design and suggests that the target function "to charge the drum" can also be performed by the transfer charger, of which the original function is "to transfer toner to the output paper." In Figure 15, black rectangular nodes perform the target function "to charge the drum" in the normal state of the copier. When the target function is lost, a potential function of the transfer charger, which is performed by the hatched rectangular nodes, can replace the target function. The hatched function hierarchy represents function redundancy.

We have developed a prototype copier that has this function redundancy (Umeda et al., 1992) (see Fig. 16). We only

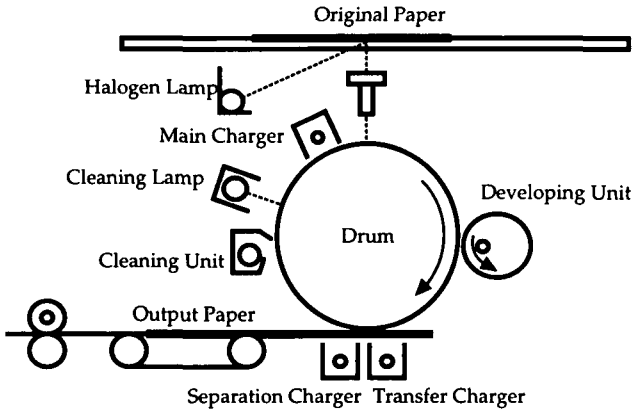

Fig. 14. Structure of a copier.

needed to replace a transformer for the transfer charger and to install an additional control sequence program. If we wanted to achieve the same performance by part redundancy, we should have needed to add an additional main charger, a transformer, and so on. This means that function redundancy allows more fault tolerance with a much smaller number of additional parts.

6. DISCUSSIONS AND RELATED WORKS

The main focus of the FBS Modeler is to support functional design not only in the analytical phase but also in the synthetic phase. We consider that the difference between analytical tasks (e.g., diagnosis [Abu-Hanna et al., 1991; Bradshaw & Young, 1991] and design verification [Iwasaki et al., 1993]) and synthetic tasks (e.g., design [Franke, 1991; Welch & Dixon, 1992]) is critical. While an analytical task is a process that transforms structural description into functional description understandably for the user, a synthetic task is a process that transforms functional description representing the user's intention into structural description. For supporting the synthetic phase of design, the FBS Modeler has two advantages; namely, its knowledge representation and the subsystem QPAS. Concerning the former, besides the mapping between functions and behaviors, which is common for existing function reasoning tools (e.g., Iwasaki et al., 1993), the FBS Modeler has two additional kinds of knowledge:

- the functional decomposition knowledge used for constructing functional description of design objects, and
- physical features used for generating behavior and structure of design objects that can perform functions. Especially, in mechanical domain, the physical feature is a more appropriate building block of behavior than the component that is often used as the building block in traditional representation, because, as Faltings (1987) pointed out, a function in mechanical domain corresponds to an interaction among some components rather than a component.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

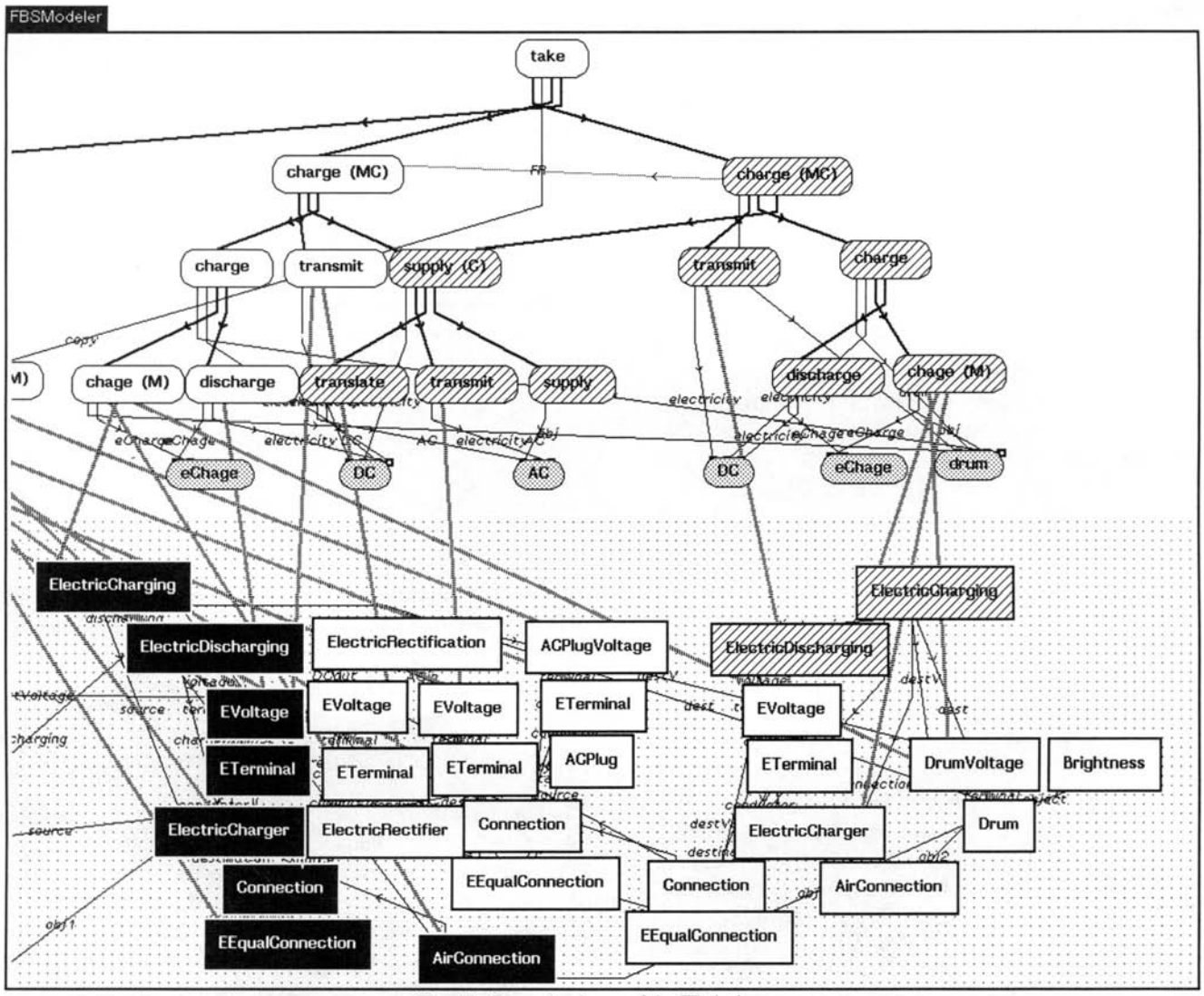

Fig. 15. Screen hardcopy of the FR designer.

We view that these two kinds of knowledge are indispensable for supporting the designer to create new design objects by selecting and combining them. Because the QPAS system assists the designer to find out new combinations of physical features, this system is also useful for creative design. The design for function redundancy described in Section 5 is an example of such creative design with the FBS Modeler, which could not be supported by the traditional CAD systems because function plays the crucial role in this design. Designers' comments described in Section 4 were also positive for this ability of the FBS Modeler to support creative design.

Fig. 16. The result of repair execution.

The classification of the functional decomposition clarifies the difference between the required functions and the functions for explanation. In other words, while the task-decomposed functional hierarchy can be considered as a description of the designer's intention, the extracted functions by the causal decomposition can be considered as additional functions for explaining the mechanism to realize the designer's intention.

One of the limitations of our approach is that, presently, the FBS Modeler deals with behaviors corresponding only to the undecomposable level of the function hierarchy. Allemang and Liver (1994) proposed the idea of *horseshoe constraint* as the behavioral descriptions of abstract functions.

https://doi.org/10.1017/S0890060400001621 Published online by Cambridge University Press

Keuneke (1991) categorized four function types; namely, ToMake, ToMaintain, ToPrevent, and ToControl, while we do not. Iwasaki et al. (1993) represented behavioral functions based on causality among state transitions. This approach is similar to ours with different knowledge representation and less emphasis on design. Bracewell et al. (1995) proposed an integrated CAD tool in which functions are embodied as bond graphs. This approach is effective for the domain where design objects can be represented well with the bond graphs. Welch and Dixon (1992) also proposed a bond graph based functional representation. Our approach agrees with Keuneke's approach in that functional structure does not always correspond to physical structure, while some others (e.g., Pahl & Beitz, 1988; and Bradshaw & Young, 1991) assume so. We further demonstrate the importance of this with the function redundancy in Section 5.

Function redundancy is a reverse operation of the *function sharing* (Ulrich & Seering, 1988) to obtain higher reliability. Comparing the design for function redundancy with Suh's axiomatic design (Suh, 1990), the design for function redundancy does not fit to the independence axiom. This is because the design for function redundancy focuses on robustness against faults of the design object, while the axiomatic design focuses on the traditional performance.

7. CONCLUSIONS

In this paper, we have proposed the FBS Modeler, which supports designers not only in the analytical phase of conceptual design but also in the synthetic phase. The functional decomposition knowledge and the physical feature in the knowledge representation of the modeler and the QPAS enable the modeler to support the synthetic phase. We have confirmed the advantages of the FBS Modeler with experimental use of the modeler by designers and demonstrated the usefulness of our approach with the design of functional redundancy.

Future work includes:

- Verifying applicability of the FBS Modeler to actual design.
- Supporting quantitative functional evaluation and modification of design objects based on this evaluation. We are developing a quantitative evaluation method of function called *Amount of Function*, which uses the modifiers of function symbols and a design process model called *Functional Evolution Model* (Shimomura et al., 1995).

ACKNOWLEDGMENTS

The work described in Section 4 was part of the international collaborative research project, GNOSIS (knowledge systematization), in the Intelligent Manufacturing Systems (IMS) research program. The examples of house design and representation of products were developed under the collaboration with Shimizu Construction Co., Ltd. and Mitsubishi Electric Co., Ltd., respectively.

REFERENCES

Abu-Hanna, A., Benjamins, R., & Jansweijer, W. (1991). Device understanding and modeling for diagnosis. *IEEE Expert 6(2)*, 26–32.

Allemang, D., & Liver, B. (1994). A functional representation for design. In *Workshop Notes of Reasoning About Function, AAAI-94 Workshop Program*, (Hodges, J., Ed.), pp. 9–19. AAAI, Menlo Park, CA.

Bracewell, R.H., Langdon, P.M., Oh, W.K., Chaplin, R.V., Li, M., Yan, X.T., & Sharpe, J.E.E. (1995). Integrated platform for AI support of complex design—(part I): Rapid development of schemes from first principles. In *Preprints of the First IFIP WG 5.2 Workshop: Knowledge Intensive CAD-1*, (Tomiyama, T., Mantyla, M. and Finger, S., Eds.), pp. 263–281. IFIP.

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

Faltings, B. (1987). Qualitative kinematics in mechanisms. *Proc. IJCAI-87*, 436–442.

Forbus, K. (1984). Qualitative process theory. *Artif. Intell. 24(3)*, 85–168.

Franke, D.W. (1991). Deriving and using descriptions of purpose. *IEEE Expert 6(2)*, 41–47.

Ireson, W.G. (1966). *Reliability handbook.* McGraw-Hill, New York.

Ishii, M., Tomiyama, T., & Yoshikawa, H. (1993). A synthetic reasoning method for conceptual design. *IFIP World Class Manufacturing '93*, pp. 3–16. Amsterdam, North-Holland.

Iwasaki, Y., Fikes, R., Vescovi, M., & Chandrasekaran, B. (1993). How things are intended to work: Capturing functional knowledge in device design. *Proc. IJCAI'93*, pp. 1516–1522.

Keuneke, A.M. (1991). Device representation: The significance of functional knowledge. *IEEE Expert 6(2)*, 22–25.

Miles, L.D. (1972). *Techniques of value analysis and engineering.* McGraw-Hill, New York.

Pahl, G., & Beitz, W. (1988). *Engineering design: A systematic approach.* Springer-Verlag, Berlin.

Rodenacker, W. (1971). *Methodisches Konstruieren.* Springer-Verlag, Berlin.

Shimomura, Y., Takeda, H., Yoshioka, M., Umeda, Y., and Tomiyama, T. (1995). Representation of design object based on the functional evolution process model. In *9th Int. Conf. Design Theory and Methodology—DTM '95*, (Ward, A.C., Ed.), pp. 351–360. ASME, New York.

Suh, N.P. (1990). *The principles of design.* Oxford University Press, New York.

Tomiyama, T., Kiriyama, T., & Umeda, Y. (1994). Toward knowledge intensive engineering. In *Knowledge Building and Knowledge Sharing*, (Fuchi, K. and Yokoi, T., Eds.), pp. 308–316. Ohmusha and IOS Press, Tokyo.

Tomiyama, T., Kiriyama, T., & Yoshikawa, H. (1992). Conceptual design of mechanisms: A qualitative physics approach. In *Concurrent Engineering: Automation, Tools, and Techniques*, (Kusiak, A., Ed.), pp. 131–152. John Wiley & Sons, New York.

Ulrich, K.T., & Seering, W.P. (1988). Function sharing in mechanical design. *Proc. AAAI-88*, pp. 342–346.

Umeda, Y., Takeda, H., Tomiyama, T., & Yoshikawa, H. (1990). Function, behaviour, and structure. In *Applications of Artificial Intelligence in Engineering, V*, (Gero, J.S., Ed.), pp. 177–193. Computational Mechanics Publications and Springer-Verlag, Southhampton and Berlin.

Umeda, Y., Tomiyama, T., & Yoshikawa, H. (1992). A design methodology for a self-maintenance machine based on functional redundancy. In *Design Theory and Methodology—DTM '92*, (Taylor, D.L. and Stauffer, L.A., Eds.), pp. 317–324. ASME, New York.

Welch, R.V., & Dixon, J.R. (1992). Representing function, behavior and structure during conceptual design. In *Design Theory and Methodology—DTM '92*, (Taylor, D.L. and Stauffer, L.A., Eds.), pp. 11–18. ASME, New York.

Yoshikawa, H., & Gossard, D., Eds. (1989). *Intelligent CAD, I.* North-Holland, Amsterdam.

Yasushi Umeda has been a lecturer at the Inverse Manufacturing Laboratory, Graduate School of Engineering, the University of Tokyo since 1995. He received a doctoral degree in precision machinery engineering from the Graduate School of the University of Tokyo in 1992. His research in-

terests include intelligent CAD (especially for conceptual design), functional reasoning, green life cycle design, and soft machines (self-maintenance machines and cellular machines).

Masaki Ishii is a Ph.D. candidate at the Department of Precision Machinery Engineering, the University of Tokyo. He received his B.Sc. and M.Sc. degrees in precision machinery engineering from the University of Tokyo in 1991 and 1993. His current research interests include knowledge sharing, model-based reasoning, knowledge intensive engineering, and large-scale engineering knowledge bases.

Masaharu Yoshioka has been a Research Associate at the National Center for Science Information Systems (NACSIS) in Tokyo, since 1996. He received a doctoral degree in precision machinery engineering from the Graduate School of the University of Tokyo in 1996. His research interests include design process modeling, functional modeling, and knowledge intensive engineering.

Yoshiki Shimomura has been a researcher at Mita Industrial since 1988. He graduated from the Kyushyu Institute of Technology in 1984. His research interests include functional reasoning and representation and intelligent mechanical systems. He has participated in the Self-Maintenance Machine Project at the University of Tokyo since 1988 and developed self-maintenance copiers as products of Mita Industrial.

Tetsuo Tomiyama has been an Associate Professor at the Department of Precision Machinery Engineering, the University of Tokyo, since 1987. From 1985 to 1987, he worked at the Centre for Mathematics and Computer Science in Amsterdam. He received a doctoral degree in precision machinery engineering from the Graduate School of the University of Tokyo in 1985. His research interests include design theory and methodology, knowledge-intensive engineering, applications of qualitative physics, large-scale engineering knowledge bases, soft machines (self-maintenance machines and cellular machines), and manufacturing paradigms.