Development of design methodology for upgradable products based on function–behavior–state modeling

YASUSHI UMEDA,¹ SINSUKE KONDOH,² YOSHIKI SHIMOMURA,³ AND TETSUO TOMIYAMA⁴

¹Department of Mechanical Engineering, Graduate School of Engineering, Osaka University, Yamada-Oka 2-1, Suita, Osaka 565-0871, Japan

²National Institute of Science and Technology, Tsukuba, Japan

³Faculty of System Design, Tokyo Metropolitan University, Tokyo 192-0397, Japan

⁴Faculty of Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

RECEIVED May 23, 2004; ACCEPTED January 24, 2005

Abstract

Extending product life is one of the hopeful approaches to reduce the environmental issue, which is one of the most critical issues of today. However, many products are thrown away because of obsolescence of functions and their performance. Therefore, we should design products to be functionally upgradable. Moreover, such upgradable products may create business chances at later stages of product life cycles. The objective of this research is to propose a design methodology for upgradability. This methodology employs a functional modeling scheme, FBS modeling, because upgrade design is a distinctive application of functional design that aims at maximizing functional flexibility with minimal structural changes after the product is manufactured. Here, the functional flexibility refers to an ability of a product to adapt its functions to changes of user needs. This paper proposes and models design processes and design operations in the upgrade design. Especially, the methodology supports finding out candidates of modifications of the function structure and configuration of a platform, which is common structure of a product among several generations, and upgrade modules. One of its central issues of upgrade design is treatment of future uncertainty. For this purpose, we propose two design strategies: delayed selection of components, and expanding and shrinking platform. A prototype system and a case study of upgrade design for a vacuum cleaner are also illustrated. The case study indicates that the system succeeded in systematically supporting a designer to execute the design methodology. Regarding the functional design, as an extension of FBS modeling, this paper proposes a method to relate abstract entity concepts in FBS modeling to concrete components through a quantitative behavior model and range calculation, in addition to deployment of FBS modeling for the design methodology.

Keywords: Design Methodology; Environmentally Conscious Design; Function–Behavior–State Modeling; Upgrade Design; Vacuum Cleaner

1. INTRODUCTION

The environmental issue is one of the most critical issues in the present time. This problem is mainly caused by the current mass production of artifacts. The mass production inevitably makes products' lifetimes shorter to increase product sales, and this results in excessive consumption of natural resources and generation of wastes (Tomiya, 1997). *Design* plays a crucial role in solving the environmental

issues, because design can determine most properties of a product life and, therefore, a designer can design a product that drastically reduces emissions, natural resource consumption, and waste amount throughout a product life cycle. For example, although it is difficult and expensive to recycle a product that is not designed to be recycled, design for recyclability significantly eases recycling of a product. For this purpose, various design methodologies have been proposed; examples include design for disassembly (Jovane & Alting, 1993), design for recyclability (Hiroshige et al., 2001), end of life design (Ishii, 1999), and life-cycle planning (Kobayashi, 2001).

Extending product life is one of the hopeful approaches to reduce the environmental issues. This has been the cen-

Reprint requests to: Yasushi Umeda, Department of Mechanical Engineering, Graduate School of Engineering, Osaka University, Yamada-Oka 2-1, Suita, Osaka 565-0871, Japan. E-mail: umeda@mech.eng.osaka-u.ac.jp

tral topic of maintenance engineering and design for reliability. However, there is not only physical lifetime, which the reliability theory deals with, but also value lifetime that terminates the life of a product; a product is thrown away because of obsolescence of functions and performance (e.g., computers and cellular phones), although the product might work well. Therefore, to extend product life, we should design products that have functional upgradability in addition to physically longer life. This will result in reducing both resource (material and energy) consumption and waste generation. At the same time, such upgradable products may create new business chances at later stages of a product life cycle, such as upgrading services, reuse of products, and remanufacturing of products (Tomiyama, 1997).

Upgrading is commonly practiced in software and highly modularized machines (e.g., computers). In the domain of machine tools, “retrofit,” which is a kind of upgrading, is practiced. However, design methodology for such upgradable products (let us call it *upgrade design*) has hardly been discussed, while similar ideas have been intensively discussed in the research area of product platform design (e.g., Fujita et al., 1999; Martin & Ishii, 2002). We will discuss the difference between upgrade design and product platform design in Section 3. Some advanced companies in manufacturing environmentally conscious products (e.g., Fuji Xerox, Japan) actually have general guidelines that include requirements for upgradability, such as to extend parts’ life, to use standard parts, and to make parts replaceable. However, because designers design them in a trial and error manner, based on these guidelines, a systematized design methodology for upgradable products is strongly needed, especially in these companies.

The objective of this research is to propose a design methodology for upgradable products and to develop a design support system for upgradability. Here, because “upgrading” means changes of functions and their performance of a product after the product is shipped, upgrade design is a subproblem of functional design, of which the objective is to maximize functional flexibility with minimal structural changes. Here, functional flexibility refers to an ability of a product to adapt its functions to changes of user needs, because the manufacturing flexibility refers to the ability of a manufacturing system to adapt to changes of manufacturing needs. Therefore, this paper discusses the design methodology for upgradability as an engineering application and a methodological extension of the functional design, focusing on representation and manipulation of functions.

The rest of this paper is organized as follows. Section 2 briefly introduces a functional modeling scheme, function–behavior–state (FBS) modeling (Umeda et al., 1996), which we employ as the functional modeling infrastructure. Section 3 clarifies framework of the design methodology, with comparison of the product platform design, and introduces some basic approaches to develop the design methodology. Section 4 proposes and formalizes the process of the design methodology for upgradability based on the discussions in

Section 3. Section 5 describes a prototype system and illustrates a case study, which makes a vacuum cleaner upgradable by using the prototype system. Section 6 discusses feasibility and issues of the proposed design methodology, and Section 7 concludes the paper.

2. FBS MODELING

For supporting upgrade design, representing and manipulating functions are essential, as discussed in the previous section. In this section, as a preparation for discussing the design methodology for upgradability, we briefly introduce FBS modeling (Umeda et al., 1996), which we employ as the functional modeling infrastructure.

Various approaches for supporting functional design have been proposed (Umeda & Tomiyama, 1997). Sembugamoorthy and Chandrasekaran (1986) are pioneers in functional modeling. Prabhakar and Goel (1998) proposed a structure–behavior–function (SBF) model and applied it to analogical design. Qian and Gero (1996) also proposed a functional modeling method applied to analogical design. They extended their approach to the “function–behavior–structure” framework¹ (Suwa et al., 1999; Gero & Kannengiesser, 2004). Chakrabarti and Bligh (2001) proposed a methodology for supporting conceptual design that can deal with geometric configuration of mechanical devices. Bracewell and Sharpe (1996) developed Schemebuilder, which supports functional design based on Bond Graph representation. Among others, we here employ FBS modeling because it directly represents mappings among function, behavior, and product structure, which is an indispensable feature for the design methodology for upgradability, and supports the embodiment process of a function.

2.1. Representation of function in FBS modeling

FBS modeling defines a function as “a description of behavior abstracted by human through recognition of the behavior in order to utilize it” (Umeda et al., 1990), and represents it as an association of two concepts: symbol of human intention represented in the form of *to do something*, and behavior that can exhibit the function. Although the symbol is meaningful only to designers, this information, associated with its behavior, is essential for supporting design such as reuse of design results. Figure 1 represents the relationship among function, behavior, and state. In this figure, a function–behavior (F-B) relation, which associates a function symbol with a set of behaviors, represents a function. These F-B relations are subjective and many–many correspondent. This feature of F-B relations is essential for making upgrade design flexible. In contrast, behavior is defined as a transition of states along time and a behavior–state

¹Note that FBS modeling is different from their “function–behavior–structure” framework.

Fig. 1. The relationship among function, behavior, and state (adapted from Umeda et al., 1996).

(B-S) relation relates a behavior to states by using physical knowledge. Here, a state is represented by entities, their attributes, and their structure.

2.2. FBS modeler

A functional design support system called “FBS modeler” (Umeda et al., 1996) implements the framework described in Section 2.1. Figure 2 illustrates a part of a design object model in the FBS modeler, of which the main function is “to charge drum,” which often appears in the design of photocopiers. The upper half of this figure represents a function hierarchy of this design object; for instance, the main function “to charge drum” is decomposed into functions “to rotate drum” and “to discharge voltage to drum.” The lower half of this figure represents behavior and states of the design object; for example, the function “to discharge voltage to drum” is performed by behaviors “electrical charging” and “electrical discharging” that occur on two entities “Drum” and “Discharger,” which are related with each other by a relation “Connection With A Space.” Although the function hierarchy is subjectively constructed by a designer, the representation of the behavior network, the lower half of the figure, is based on a Qualitative Process Theory (Forbus, 1984), and its consistency is managed by a computer environment, Knowledge Intensive Engineering Framework (KIEF; Yoshioka et al., 2004). Please refer to Yoshioka et al. (2004) for the representation and reasoning of the behavior network.

To support functional design in the FBS modeler, the modeler has two kinds of knowledge bases: a function prototype knowledge base, and a physical feature knowledge base. We have verified that these knowledge bases can be constructed reasonably (Umeda et al., 1996; Yoshioka et al., 2004). A function prototype in the former knowledge base represents a typical pattern of functions that often appears in a certain domain. Table 1 shows the scheme of the function prototype. *Name* is the symbol for representing the

Table 1. Definition of function prototype

Item	Contents
Name	Symbol in the form of “to do something”
Decomposition	Networks of subfunctions
Function–behavior relations	Physical features

Adapted from Umeda et al. (1996).

designer’s intention. *Decomposition* describes feasible candidates for decomposing the function in the form of networks of subfunctions. The *F-B Relations* describe candidate embodiments of the function in the form of *physical features* to be described below.

A *physical feature* is a building block consisting of entities and physical phenomena occurring on the entities (Tomiyama et al., 1992). An FBS modeler represents a physical feature as a network of three kinds of elements (we call them *behavior nodes*); entities, relations among entities, and physical phenomena (Yoshioka et al., 2004). Each physical feature is constructed by the designer in order to be a meaningful block for representing a function. For example, Figure 3 shows an example of the physical feature in which a phenomenon “Electrical Discharging” occurs between a discharger and a nonconductor related by the relation “Connection With A Space.”

3. FRAMEWORK OF DESIGN METHODOLOGY FOR UPGRADABILITY

In this section, we clarify the framework of the design methodology for upgradability proposed in this paper and introduce some basic approaches to develop the design methodology.

Upgrading means manipulating a functional configuration of a product after the product is sold to a customer in order to adapt to changes of customer’s needs. Manipulation of the functional configuration includes adding, removing, and replacing functions (e.g., adding a two-sided copying function to a photocopier), and increasing and decreasing performance of particular functions (e.g., increasing copying speed). Such upgrade manipulation might be executed as a part of maintenance of a product or remanufacturing of a disposed product. Upgrade design is to design a product in order to add upgradability. Some products could be upgraded even though they are not designed to be so. However, upgrade design is expected to realize more flexible and more systematic upgradability of products.

3.1. Two assumptions for the design methodology for upgradability

To formalize upgrade design, we introduce two assumptions. Although there might be some other upgrade design

- : function node
- : objective node
- : physical phenomenon
- : entity, relation
- : super-sub relation between functions
- : function-objective relation
- : F-B relation
- ← : physical dependency between behavior nodes

Fig. 2. The FBS modeler (adapted from Umeda et al., 1996).

Fig. 3. An example of a physical feature (adapted from Umeda et al., 1996).

methodologies that have no upgrade plan or no platform, this paper deals with a methodology with these assumptions.

First, the design methodology only deals with functions specified to be upgraded at the time of design. Without this assumption, the design methodology has to be able to design a product that can perform functions unknown at the time of

design; namely, an almighty machine. This is impossible. For this purpose, we assume that a designer should determine a long-term plan (let us call it an *upgrade plan*) of upgrading among several generations (e.g., three generations) at the very beginning of upgrade design. As discussed later, the upgrade plan inevitably includes uncertainty, and its treatment is one of the central issues of the design methodology.

Second, we assume that upgrading of a product is executed by changing values of attributes in the product or exchanging modules (we call these *upgrade modules*). Although there might be other measures, such as remodeling a product, we do not take them into account. Therefore, a design solution of the design methodology, which represents several generations of a product, consists of a *platform*, which is a common structure of the product that does not change throughout all generations, and upgrade modules (see Fig. 4). Therefore, determining size of the platform is another central issue of the design methodology.

Fig. 4. The framework of design of upgradability.

Based on these assumptions, the design methodology for upgradability is modeled as a mapping from an upgrade plan to a series of several generations of a product (see Fig. 4). In other words, the design methodology is to design a series of products that realizes a given upgrade plan with minimum total costs (environmental and monetary) of the series as a whole and maximum flexibility against uncertainty of the upgrade plan. In this sense, the design methodology essentially contains a trade-off problem (costs vs. flexibility) and strategic decisions of a designer are indispensable; in other words, the design methodology aims at providing a design workspace to embody the upgrade plan according to his or her decisions.

The second assumption immediately recalls various studies of product platform design and design for variety. Ulrich (1995) is one of the pioneers in product architecture design. In his definition, the product architecture includes arrangement of functional elements, mapping from functional elements to physical components, and specification of interfaces among components. Fujita et al. (1998, 1999; Fujita, 2002) proposed optimization methods for finding out modules to be commonized among multiple products in order to minimize the total costs. Martin and Ishii (2002) proposed a quality function deployment based methodology for guiding construction of a product family by introducing a generational variety index and a coupling index with guiding rules to reduce values of these indices. Simpson et al. (2001) proposed a design methodology for scalable product platforms. They extend this method to select common and scaling parameters by introducing product family penalty function (Messac et al., 2002). Li and Azarm (2002) evaluated advantages of product line design in the context of customers' preferences, market competition, and multiple business goals when uncertainties of these factors exist.

Upgrade design can be viewed as a variant of platform design; we can point out three issues inherent in upgrade design. First, in many existing studies, function structure is fixed and common among a product family. In contrast, each generation, which corresponds to a product in a product family, has different function structure in upgrade design. This requires explicit representation and manipulation of functions in the design methodology. Second, alternatives for a module are given or the units of modules are predetermined in many studies. However, changes of function structure among generations inevitably require determination of boundaries of modules and alternatives for each module in the design methodology. This paper assumes that a module consists of one or more components and a lot of alternatives for each kind of components are stored in a component database. In comparison with Ulrich's definition of product architecture (Ulrich, 1995), our design methodology tries to solve, in the context of upgrade design, the problems of the arrangement of functional elements and the mapping from functional elements to physical components. These processes were not addressed well in the existing studies. Under this problem setting, we especially focus on candi-

date generation processes of changes of function structure and configurations of a platform and modules rather than evaluation and optimization of given candidates, because candidate generation itself is an issue essential for design; then, various existing evaluation and optimization methods (e.g., Fujita et al., 1999; Umeda et al., 2000; Li & Azarm, 2002) can be employed.

3.2. Upgrade plan

As shown in Figure 4, a designer should first determine a long-term upgrade plan. To make the upgrade plan the designer should consider technological trends, trends of customers' needs, trends of competitors, the company's policy, and so on, during several generations of the target product. The plan should include functions to be upgraded, basic specification, rough price of each generation, and the time when each generation will be shipped to the market. Candidates of target functions to be upgraded include functions that will be technically obsolete and appearance that will be out of fashion.

Of course, because the upgrade plan forecasts future trends of the target product, it includes uncertainty as shown in Figure 4. If the plan will not fit the actual trends after designing, the company may suffer a great loss. This is one of the main issues of the design methodology and our approach to this issue is described in Section 3.3. Moreover, we believe that there are product categories, including photocopiers, personal computers, and machine tools, for which upgrade plans can be determined with some certainty.

Although upgrade planning is a difficult task, this paper does not support the upgrade planning process. We are currently developing methods for supporting this process, including a method for identifying functions to be upgraded based on reasons why a product is thrown away (Umeda et al., 2003), a method for estimating when a function will be obsolete (Daimon et al., 2004), and a method for determining value ranges of upgrading parameters based on technological trends of components (Matsuda et al., 2003). Because these methods are not mature yet, let us assume that an upgrade plan with some uncertainty is given.

3.3. Basic approaches to design methodology for upgradability

For supporting the upgrade design process that starts from a given upgrade plan and a given design object model of the first generation of a target product, we take the following approaches.

3.3.1. Two guiding principles for incorporating upgrading functions

We have proposed two guiding principles for incorporating upgrading functions into a product; *making functions independent* and *making functions insensitive*. We derived

them from our analyses of practical design data and experimental upgrade design (Umeda et al., 1999).

Making functions independent means to localize structural changes necessary for upgrading functions. Modularization of related components and exchanging modules is a hopeful approach for implementing this principle. For example, to increase copying speed of a photocopier, modularization of the main motor and replacement of motors is a practical solution. Making functions insensitive means absorbing the difference between before and after upgrading without structural changes by adding margins to components. It suggests using components of which functionality and performance are excessive to the first generation but indispensable for subsequent generations. This strategy tries to add flexibility of performance without replacing components by adding margins to components in advance. In the above example of the photocopier, if the designer uses a better motor in the previous generation, copying speed can be increased just by controlling the motor in the next generation. This is a typical example of this principle.

3.3.2. Functional upgrading and parametric upgrading

The upgrading operation can be classified into two kinds: *functional upgrading*, which adds or removes functions such as adding the two-sided copying function to a photocopier, and *parametric upgrading*, which changes the performance of a product such as increasing copying speed. Although functional upgrading results in changes of the function structure and physical configuration of a product, parametric upgrading does not change them to a large extent. We propose different design procedures for them.

We model functional upgrading as the task to find the embodiment of each upgrading function (viz., function structure, behavior, and physical structure) and to merge this embodiment into the target product. Here, this additional embodiment does not always become an upgrade module; that is, a part of embodiment might be installed into the platform at the time when the product is manufactured as a result of the making functions insensitive principle and the rest becomes an upgrade module. Therefore, determining boundaries of the platform and upgrade modules is an important issue in functional upgrading as this affects costs and flexibility of upgradable products.

Parametric upgrading consists of the determination of parametric changes and selection of components. The task

of determination of parametric changes is to find out an appropriate set of value changes of design parameters that can realize required performance changes such as copying speed in the previous example. Here, a design parameter denotes a parameter that a designer can directly determine such as speed of a motor and dimensions of a part. Next, a designer should decide how to realize the changes. Replacing components (e.g., replacing a motor with another high speed motor) is one way to realize such changes and using a controllable component (e.g., variable resistance) is another way. The task of selection of components is to find a set of replaceable components or a controllable component that realizes derived value changes of design parameters. For this purpose, we prepare a database of components. In this way, we solve the problem of parametric upgrading by using a similar technique to those of the existing platform design.

Table 2 summarizes the correspondence between the two types of upgrade design discussed here and the two guiding principles discussed in Section 3.3.1.

3.3.3. Dealing with future uncertainty

One of the central issues of upgrade design is how to deal with the future uncertainty that upgrade plans inevitably include. Our objective here is to provide a design methodology that supports a designer to systematically make a design object as flexible as possible against the uncertainty specified in the upgrade plan. We employ two approaches to solve this problem: *expanding and shrinking platform* and *delayed selection of components*.

For dealing with the uncertainty in the functional upgrading, we represent this uncertainty as plausibility (viz., high, medium, and low) of each upgrading function in the upgrade plan. In other words, a designer believes firmly that customers require an upgrading function with high plausibility in subsequent generations, but he or she is not sure whether the one with low plausibility is required or not. Our strategy here is to determine boundaries between a platform and upgrade modules according to this plausibility (let us call this operation expanding and shrinking platform). Namely, components that perform upgrading functions with high plausibility should be integrated into the platform as much as possible, to ease upgrading operation. Conversely, components for functions with low plausibility should be isolated from the platform as upgrade modules, in order to

Table 2. Upgrade design operations and guiding principles

	Parametric Upgrading	Functional Upgrading
Making functions independent	Value changes of design parameters by replacing components	Modularizing embodiment of upgrading functions
Making functions insensitive	Value changes of design parameters by actuating controllable components	Burying a part of embodiment of upgrading functions into the platform

avoid the risk that those functions are not required in subsequent generations.

For parametric upgrading, we represent uncertainty in an upgrade plan as ranges of performance values of a function; namely, we assume that a designer does not know an exact value of performance in the next generation, but knows a certain range within which required upgrade performance resides. The parametric upgrading in this research converts these required ranges into a set of components, which are the same kind but different in performance, and/or a controllable component that can cover the required ranges. At the time of designing, a designer just selects the set of components, and when upgrading is executed, he or she chooses the most appropriate performance value from the designed range and this results in selecting a component from the set of components. Therefore, we call this operation delayed selection of components. This idea is quite similar to the postponement in the area of supply chain management (e.g., Johnson & Anderson, 2000), and this indicates that the idea of postponement can be applied to the upgrade design.

4. PROCESS OF DESIGN METHODOLOGY FOR UPGRADABILITY

Based on the discussion in Section 3, we here propose the process of the design methodology (see Fig. 5). Note that actual design is not as straight forward as this and includes a lot of trial and error, but such local loops are ignored here for the sake of simplicity.

The process shown in Figure 5 assumes an upgrade plan and an initial model of a target product, which has no upgradability, as inputs. An upgrade plan includes requirements of the functional upgrading and the parametric upgrading. An initial model consists of an FBS model of the design object, described in Section 2, a list of components in the design object, and a set of simultaneous equations representing quantitative behavior of the design object, which are used in the parametric upgrade design described below. Note that this methodology focuses on deriving candidates of the following three items, which will be evaluated with existing methodologies of, for example, platform design; namely, changes of function structure for realizing the functional upgrade, configurations of a platform and upgrade modules, and sets of concrete components materializing entities, which represent classes of components in the FBS model.

Let us take an example of a vacuum cleaner for explaining the design process, of which component configuration and simultaneous equations of the initial model are shown in Figure 6. In addition, let us assume an upgrade plan shown in Table 3 is given. For explaining the following process, we mainly focus on the upgrading function “to brush out dust” in the plan, which should be added with high plausibility in the second generation.

4.1. Constructing FBS fragments

As the first step of the process, a designer should find the embodiment (function structure and its physical realiza-

Fig. 5. The design process for upgradability.

Fig. 6. The initial model of the vacuum cleaner.

tion) of each upgrading function in each generation. This task is almost the same as the main task of the traditional functional design; namely, the embodiment process of required functions. However, the designer should determine candidates of embodiments by referring to the initial FBS model (let us call it *main body*) in order to be easily integrated. Because the FBS modeler supports the embodiment process, we employ the FBS modeler. As an extension of the modeler, the system supports to construct a *partial* FBS model that realizes a required upgrading function (let us call this partial FBS model an *FBS fragment*), rather

than a *complete* FBS model. An FBS fragment is different from the complete FBS model in the point that the FBS fragment may lack some subfunctions, behaviors, or entities by assuming that the fragment will be connected to the main body of the design object. Let us call them *void* functions, *void* behaviors, and *void* entities. Void nodes store pointers to the corresponding nodes in the main body.

In other words, this step is formalized as Eq. (1):

$$f_u \rightarrow \langle F_{fr}, B_{fr}, E_{fr}, R_{fr} \rangle, \quad (1)$$

Table 3. Example of upgrade plan

		Generation		
		1st	2nd	3rd
Parametric upgrading	Suction power (W)	240	300–360	360–480
	Power consumption (W)	640	640–700	700–800
	Noise (dB)	60	55–60	50–55
	Cleanness of exhaust air (%)	98	98–99	99–99.5
Functional upgrading	Addition of function	—	To brush out dust (H)	To charge electricity (H)
	Deletion of function	—	—	To store electric cord (L)

H and L, high and low plausibility of upgrading functions, respectively.

Fig. 7. The function decomposition for an upgrading function "to brush out dust."

where f_u is the target upgrading function, F_{fr} is a set of functions composing an FBS fragment, and $f_u \in F_{fr}$, some of which might be void functions; B_{fr} is a set of behaviors composing an FBS fragment and some of them might be void behaviors; E_{fr} is a set of entities composing an FBS fragment, some of which might be void entities; and R_{fr} is a set of relations composing an FBS fragment.

The methodology supports the designer to construct FBS fragments as described in the following sections.

4.1.1. Function decomposition

The designer breaks down the upgrading function into subfunctions repeatedly by referring to the function prototype knowledge base of the FBS modeler. Different from construction of the complete FBS model, the designer may specify a subfunction as a *void* function, if he or she finds the same kind of function in the main body and judges that the FBS fragment can be embodied by using this existing function. The prototype system, described in Section 5.1, supports this task by searching for the same kind of functions throughout the main body. A void function is not further decomposed nor embodied.

Figure 7 illustrates an example of function decomposition of an FBS fragment for the function "to brush out dust." In this example, the designer decomposes this function into two subfunctions and then decomposes a subfunction "to rotate object" into three by choosing a decomposition method "electrical power rotation method," which is prepared in the knowledge of function prototype "to rotate object." Here, the designer sets a subfunction "to supply

electricity" as a void function and, therefore, this function is not further decomposed nor embodied.

4.1.2. Selection of physical features

After decomposing the upgrading function into element functions, the designer determines the physical embodiment that can realize each elementary function by selecting appropriate physical features, introduced in Section 2.2. The system supports this process by searching through the function prototype knowledge base. Similar to the void functions, the designer can select void behaviors and entities in the fragment if the main body has usable behaviors and entities. Although the system searches for the same kinds of behaviors and entities in the main body, the designer should judge their usability in the FBS fragment.

For example, Figure 8 shows the FBS fragment of Figure 7 after adding physical features. In this case² for instance, the subfunction "to touch objects to dust" is embodied by a behavior "collision," which occurs between entities "dust" and "roller" connected with each other by a relation "contact." Intuitively, this example of an FBS fragment means adding a rotational brush to the head of the vacuum cleaner in order to realize the upgrading function "to brush out dust," which will increase the sweeping ability of the cleaner.

²Of course, the entity "dust" does not appear in the function prototype and physical feature knowledge bases, because they store general knowledge. For easier understanding, the designer rewrote a general entity "object" to be "dust" in this fragment and this is a normal operation in the FBS modeler.

Fig. 8. The physical embodiment for an upgrading function “to brush out dust.”

In this way, the designer can construct various FBS fragments from one upgrading function.

4.2. Adding and removing FBS fragments

Next, the designer applies the constructed FBS fragments to the FBS model of the initial product, the main body. As the result of this step, the designer determines tentative boundaries among platform and upgrade modules. Here, we call them *tentative* boundaries because they might change later on and will be finalized in Section 4.4. This step is formalized as the following equations.

$$\text{FBS}^i, \text{FR}_+^{i+1}, \text{FR}_-^{i+1} \rightarrow \text{FBS}^{i+1} \rightarrow p^{i+1}, \text{UM}_+^{i+1}, \text{UM}_-^{i+1}, \quad (2)$$

$$\text{FBS}^i = \langle F^i, B^i, E^i, R^i \rangle, \quad (3)$$

$$\text{FR}_+^i = \{\text{fr} \mid \text{fr} = \langle F_{\text{fr}}, B_{\text{fr}}, E_{\text{fr}}, R_{\text{fr}} \rangle\}, \quad (4)$$

$$\text{FR}_-^i = \{\text{fr} \mid \text{fr} = \langle F_{\text{fr}}, B_{\text{fr}}, E_{\text{fr}}, R_{\text{fr}} \rangle\},$$

$$F_{\text{fr}}, B_{\text{fr}}, E_{\text{fr}}, R_{\text{fr}} \subset \text{FBS}^i, \quad (5)$$

$$p^i = \langle F_p^i, B_p^i, E_p^i, R_p^i \rangle, \quad (6)$$

$$\text{UM}_+^i = \{\text{um} \mid \text{um} = \langle F_{\text{um}}, B_{\text{um}}, E_{\text{um}}, R_{\text{um}} \rangle\}, \quad (7)$$

$$\text{UM}_-^i = \{\text{um} \mid \text{um} = \langle F_{\text{um}}, B_{\text{um}}, E_{\text{um}}, R_{\text{um}} \rangle\}, \quad (8)$$

$$\left(F^i - \bigcup_{\text{fr} \in \text{FR}_-^{i+1}} F_{\text{fr}} \right) \cup \bigcup_{\text{fr} \in \text{FR}_+^{i+1}} F_{\text{fr}} = F^{i+1} = F_p^{i+1} \cup \bigcup_{\text{um} \in \text{UM}_+^{i+1}} F_{\text{um}}, \quad (9)$$

$$\left(B^i - \bigcup_{\text{fr} \in \text{FR}_-^{i+1}} B_{\text{fr}} \right) \cup \bigcup_{\text{fr} \in \text{FR}_+^{i+1}} B_{\text{fr}} = B^{i+1} = B_p^{i+1} \cup \bigcup_{\text{um} \in \text{UM}_+^{i+1}} B_{\text{um}}, \quad (10)$$

$$\left(E^i - \bigcup_{\text{fr} \in \text{FR}_-^{i+1}} E_{\text{fr}} \right) \cup \bigcup_{\text{fr} \in \text{FR}_+^{i+1}} E_{\text{fr}} = E^{i+1} = E_p^{i+1} \cup \bigcup_{\text{um} \in \text{UM}_+^{i+1}} E_{\text{um}}, \quad (11)$$

$$\left(R^i - \bigcup_{\text{fr} \in \text{FR}_-^{i+1}} R_{\text{fr}} \right) \cup \bigcup_{\text{fr} \in \text{FR}_+^{i+1}} R_{\text{fr}} \subseteq R^{i+1} = R_p^{i+1} \cup \bigcup_{\text{um} \in \text{UM}_+^{i+1}} R_{\text{um}}, \quad (12)$$

$$\text{UM}_-^{i+1} = \text{FR}_-^{i+1}, \quad (13)$$

where FBS^i is the FBS model of the i th generation of the target product, consisting of functions F^i , behaviors B^i , entities E^i , and relations R^i . Especially, let us call the model in the previous generation the *main body* of the product. In addition, FR_+^{i+1} , FR_-^{i+1} are sets of additional and removal FBS fragments for constructing $i + 1$ th generation of the product and FR_+^{i+1} , FR_-^{i+1} are added to and removed from FBS^i , respectively. Although FBS fragments in FR_+^{i+1} are constructed from required upgrading functions in Section 4.1, those in FR_-^{i+1} are selected by the first operation in Table 4. In Eq. (6), p^i is the platform of the i th generation tentatively determined by the designer and UM_+^{i+1} , UM_-^{i+1} are sets of additional and removal upgrade modules for the

Table 4. Operations for adding and removing FBS fragments

Removing Upgrading Function From Main Body
Removing the operation of a function means to find an FBS fragment in the main body, of which the top node is the removing function. As a result, the derived fragment becomes a removing upgrade module. Assuming a removing function (f^*), the removing fragment $fr_- = \langle F_-, B_-, E_-, R_- \rangle$ consists of the functions, behaviors, entities, and relations connected to the f^* in the main body. However, the fragments should not include those nodes that perform functions other than f^* , so that the whole product works properly after removing the fragment.
Adding FBS Fragment to Main Body
Addition of an FBS fragment is executed almost automatically. After a designer selects a superfunction in the main body for a top function of each fragment (viz., an upgrading function), the fragment is merged into the main body by using the information of void nodes, which have pointers to corresponding nodes in the main body as described in the text.
Determining Tentative Upgrade Modules
When the platform is determined, upgrade modules are also determined. In this step, a portion of each additional FBS fragment, which remains outside of the platform, is recognized as an upgrade module. The following equation represents this relationship among an FBS fragment (fr), the platform (p), and an upgrade module (um), where $\bar{p}$ denotes the complementary set of p .
$um = fr \cap \bar{p}.$
The designer may then combine or decompose these upgrade modules, for example, for increasing replaceability or reducing manufacturing costs.

$i + 1$ th generation, respectively. These tentative upgrade modules are also determined in this step.

Equation (9) denotes that functions included in the i th model but not included in the removal FBS fragments and functions in the set of additional FBS fragments are reorganized into the platform and the upgrade modules. The same operation is applied to behaviors and entities as shown in Eqs. (10) and (11). As shown in Eq. (12), the designer might add some relations to connect FBS fragments to the main body. In determining the removal upgrade modules, the proposed methodology assumes that each removal module is equivalent to each removal FBS fragment as shown in Eq. (13), for the sake of simplicity.

Based on the “expanding and shrinking platform” strategy, plausibility of upgrading functions should be reflected in the size of the platform. Although FBS fragments for upgrading functions with high plausibility can be integrated into the platform, fragments for functions with low plausibility should be isolated as upgrade modules. To do so, the procedure for adding and removing fragments proceeds as follows:

1. Removing all upgrading functions, which are designated to be removed in the upgrade plan, from the main body, by applying the first operation defined in Table 4.

2. Adding all FBS fragments for upgrading functions with high plausibility by applying the second operation defined in Table 4. For example, in Figure 9, the fragment for the upgrading function “to brush out dust” is connected to the main body by using its void nodes and the designer added a new relation “electric connect” between entities “electrical lead” and “power source,” which is not shown in this figure.
3. Determining the tentative platform in order to extend it as much as possible. For supporting this decision, the prototype system, described in Section 5.1, indicates the following three kinds of information to the designer. The designer determines the boundary between the smallest and the largest platform by considering, for example, costs, technological trends, and connectability.
 - a. Smallest platform: apparently, the smallest platform p_{small}^{i+1} is equal to the i th platform after removing all removing fragments FR^{i+1} created by the operation (i) in Table 4 [see Eq. (14)].

$$p_{small}^{i+1} = p^i - \bigcup_{fr \in FR^{i+1}} fr. \quad (14)$$

- b. Largest platform: obviously, the largest platform is the FBS model that includes the smallest platform p_{small}^{i+1} and all additional FBS fragments FR_+^{i+1} . However, it is infeasible to assume such a large platform that needs no additional upgrade module. Therefore, as an indicator, assuming n generations as the target of the upgrade plan, we define the largest platform p_{large} as follows, where, FR_h^i ($FR_h^i \subseteq FR_+^i$) denotes a set of all additional fragments with high plausibility for i th generation ($2 \leq i \leq n$).

$$p_{large} = (FBS^1 \cap FBS_h^2) \cup (FBS_h^2 \cap FBS_h^3) \cup \dots \cup (FBS_h^{n-1} \cap FBS_h^n), \quad (15)$$

$$FBS_h^i = p_{small}^i \cup \bigcup_{fr \in FR_h^i} fr \quad (2 \leq i \leq n). \quad (16)$$

- c. Entity commonality rate: as the third indicator, for representing the size of platform, we calculate entity commonality ratio (CR) of the platform, which indicates the ratio of entities in the platform to all entities in all generations [see Eq. (17)], where E_p denotes an entity set of the platform and $|E|$ denotes the number of entity sets.

$$CR = \frac{|E_p|}{\left| \bigcup_{i=1}^n E^i \right|}. \quad (17)$$

The idea of commonality is a very popular concept (e.g., see Fujita, 2002) and this indicator is a simplified representation of this concept.

Fig. 9. An example of the addition of an FBS fragment.

For example, Figure 9 shows a part of boundaries of the smallest and the largest platform focusing on the upgrading function “to brush out dust.” In this figure, three candidates of the boundaries are shown between the smallest and the largest platform.

4. Adding FBS fragments for upgrading functions with middle and low plausibility in order not to extend the platform by applying the second operation in Table 4.
5. Determining upgrade modules by applying the third operation defined in Table 4.

In this way, at this step “adding and removing FBS fragments,” the designer applies additional and removing FBS fragments to the initial FBS model, which results in the FBS model of the second generation, and determines tentative platform and upgrade modules. With this procedure, the designer can systematically implement the “expanding and shrinking platform” strategy.

4.3. Parametric upgrade design

Next, the designer executes parametric upgrade design, focusing on the FBS model of the second generation; namely, he or she adds parametric upgradability to the target generation and then selects concrete components of the product in this generation by using the component database, considering the “delayed selection of components” strategy. The input of this step consists of a parametric upgrade plan (see the upper part of Table 3), which represents uncertainty as ranges of parameter values, a list of components in the first generation, each of which is related to an entity node in the FBS model of the first generation, and a quantitative behavior model represented as simultaneous equations (see the upper part of Fig. 6). Here, because the behavior model of the second generation may be different from the initial one because of addition and removal of fragments, the designer should manually construct the model

of the second generation by modifying the initial one. Although the behavior model can theoretically be derived from the FBS model with the KIEF system (Yoshioka et al., 2004) and we are currently studying, this issue is one of our future works.

Although a detailed algorithm is described in another paper (Umemori et al., 2001), this step proceeds as in following subsections.

4.3.1. Construction of causal relation graph

Initially, we classify parameters in the quantitative behavior model into three kinds; *function* parameters, *design* parameters, and others (see Suh, 1990). Function parameters represent overall performance of the target product (e.g., suction power and power consumption of a vacuum cleaner) and, therefore, all parameters that appear in the upgrade plan are function parameters. Design parameters are attributes of components the designer can directly determine such as dimensions of a component and torque of a motor. After the designer specifies function and design parameters in the behavior model, the system relates them according to causalities. For example, given a tiny behavior model shown in the left side of Figure 10, the system creates a causal relation graph shown in the right side of Figure 10, where f_i and d_i denote function and design parameters, respectively. In this figure, each arrow denotes that a value change of a design parameter causes a value change of a function parameter. By changing values of design parameters and calculating their effects on function parameters, the system creates the causal relation graph.

4.3.2. Searching for design parameters

Next, the system finds candidates of appropriate design parameters for changing values of each function parameter, that is, parametric upgrading. To avoid side effects that might be caused by design parameter changes, we here employ the Independence axiom of Axiomatic Design Theory (Suh,

1990). By classifying causal relations into three types as shown in the lower part of Figure 10, type I should be selected as much as possible. Here, type I is the relation where a value change of a design parameter affects only one function parameter, type C is the relation where a value change of a design parameter affects more than one function parameter, and type SI is a special kind of type C. In the lower right of Figure 10, a design parameter (d_2) affects both function parameters f_1 and f_2 . However, because a value change of f_1 can be adjusted by controlling d_1 , the relation f_2 - d_2 can be spuriously considered as type I. Based on this idea, the system reasons out all possible sets of causal relations for controlling function parameters, which preferably consist only of types I and SI. In the example of Figure 10, a set of derived causal relations consists of f_2 - d_1 (I type), f_1 - d_2 (SI type), and f_3 - d_3 (SI type). If such a set of I and SI types cannot be found, the designer should modify the behavior model by adding, removing, or replacing components and this modification is included in the upgrade modules. Conversely, the designer should select the set of relations that has minimum number of C types for avoiding unwanted changes of function parameters.

4.3.3. Range calculation using set-based theory

By using the derived sets of causal relations, the system calculates ranges of design parameters needed for the required ranges of upgrading function parameters. For this calculation, we employ the Set-Based Theory (Finch & Ward, 1995), which provides mathematical operations for calculating ranges of parameter values. The system also calculates parametric changes of function parameters other than upgrading parameters as potential side effects. Among derived sets of changes of design parameters, the designer should choose the most appropriate set that realizes all parametric upgrading and has minimum potential side effects.

4.3.4. Components selection

To realize the specified ranges of design parameters, the system finds components by searching through the component database, which stores components that might be used in the target product and their specifications. For each design parameter's range, several components, which might be variable or not, are selected in order to cover the range.

At the same time, the designer should select components corresponding to entity nodes in the additional FBS fragments determined in Section 4.2. Here, whereas an entity node specifies only a class of a component such as a motor, a component in the database represents a specific component, such as a Toshiba AC Motor 1234, with all attributive values of the component. Therefore, because an entity node in a fragment does not know its quantitative attributes, a component should be selected for each entity node in this step by using the result of the set based quantitative calculation.

As a result, the parametric upgrading is integrated into the design object model as sets of candidate components,

Fig. 10. The extraction of a causal relation graph.

and entity nodes in the additional FBS fragments are also embodied into components. Using the “delayed selection of components” strategy, uncertainty of the parametric upgrading results in a set of candidate components corresponding to an entity node and a specific component should be selected among them when the upgrading is executed. Therefore, the step in Section 4.3 is formalized as the following equations:

$$\text{PUG}^{i+1}, C^i, \text{FBS}^{i+1}, \text{EQ}^{i+1} \rightarrow C^{i+1}, \quad (18)$$

$$C^i = \{\langle e, c \rangle \mid e \in E^i, c \in C_{\text{DB}}(e) \vee c \subseteq C_{\text{DB}}(e)\}, \quad (19)$$

where PUG^i is the parametric upgrades for the i th generation specified in the upgrade plan, C^i is the component list for the i th generation, c can be a component or a set of components, $C_{\text{DB}}(e)$ is a subset of components that belong to the component class e in the component database, and EQ^i is the quantitative behavior model of the i th generation.

4.4. Determining platform

According to the results of Section 4.3, the platform and upgrade modules are adjusted. In the component list derived in Section 4.3, an entity related to a set of components, rather than to a component, cannot be included in the platform because its corresponding component is not determined at the time of manufacturing. Therefore, if the platform includes such entities, they are moved to the upgrade modules. In this way, platform and upgrade modules for the second generation are decided. However, the final platform will be determined only after all generations have been designed.

4.5. End of design for a generation

In this way, the upgrade design process for the second generation finishes and the designer should repeat the steps in Sections 4.1–4.4 for all generations specified in the upgrade plan. When the final generation is designed in this cycle, the final platform and upgrade modules are determined as the output of the design process.

5. PROTOTYPE SYSTEM AND CASE STUDY

To verify feasibility and to clarify issues of the proposed design methodology, we developed a prototype upgrade design support system and conducted an experimental upgrade design of a vacuum cleaner by using the system.

5.1. Prototype system

Figure 11 depicts the architecture of the prototype system, and Figure 12 illustrates an example of a screen dump of the prototype system. Although we employ the existing FBS modeler described in Section 2.2 and some subsystems have

not been implemented yet, the main subsystems are the following:

- UG Manager guides the designer to execute the design for upgradability.
- FBS Manager supports construction of the FBS models and the FBS fragments and to add/remove fragments to/from the FBS model, in cooperation with FBS Modeler, FBS Knowledge Bases, and FBS Fragment Modeler.
- PUG Manager supports execution of the parametric upgrading. The Causal Relation Graph Modeler constructs the causal relation graph from the quantitative behavior model and searches for appropriate sets of causal relations to control required function parameters. The Set-Based Theory Calculator calculates parameter ranges based on the Set-Based Theory, and the Components Selector finds appropriate sets of components by searching through the Components Database.
- Platform Designer supports the designer to determine a platform and upgrade modules by using the FBS model, the FBS fragments, and results of the parametric upgrading.

The prototype system represents various models and knowledge discussed in Section 4, such as the FBS model, the FBS fragment, the behavior model, and the component list, and supports user’s operations for them. Especially, the system supports the following tasks:

- Construction of the FBS model and the FBS fragment are supported by using the FBS Knowledge Bases.
- FBS fragments to be removed are created from specified removing functions by searching through the FBS model.
- Additional FBS fragments are integrated into the FBS model.
- For a specified additional fragment, the system suggests the smallest and the largest boundaries of the platform and, after the designer determines the boundary, the system indicates an upgrade module.
- To support the parametric upgrading, the system constructs the causal relation graph from a given quantitative behavior model, reasons out candidates of design parameters for controlling given upgrading parameters, calculates value ranges of design parameters, and finds appropriate component sets by searching through the component database.
- To determine the platform, the system indicates components that are included in the platform but are designated to be replaced as the result of the parametric upgrade design. Such components should move to upgrade modules.

Fig. 11. The architecture of the prototype system. SBT, Set-Based Theory.

5.2. Case study

We executed an experimental upgrade design with the example of a vacuum cleaner (see the lower part of Fig. 6). The data for this case study were acquired from our analyses of

some commercial vacuum cleaners and from interviews to engineers. The inputs for this case study consist of an upgrade plan (Table 3), the initial FBS model (Fig. 13), a quantitative behavior model (Fig. 6), and a list of components (Table 5).

Although the design results may differ according to the designer's decisions, upgrade design in this case study proceeded as follows. The upgrading function "to brush out dust" in the second generation is embodied by adding a motor and a rotational brush at the head of the cleaner as described in Figures 7–9. In the third generation, additional function "to charge electricity" and removal function "to store electric cord" aim at making the cleaner wireless and are realized by replacing the electric cord with a battery, a charger, an electric converter, and an electric inverter.

As a result, we derived four candidates for the platform and, among them, the largest platform (entity commonality rate = 56%³) and the smallest one (entity commonality rate = 41%⁴) are shown in Figure 14. Table 5 lists derived

Fig. 12. An example of a screen dump of the prototype system.

³In this case, 27 different components are used in total among the three generations and 13 components out of 27 are included in the platform. Thus, 13/27 = 56%.

⁴In this case, 11 components out of 27 are included in the platform. Thus, 11/27 = 41%.

Fig. 13. A part of the FBS model of the first generation in the FBS modeler.

Table 5. Derived component list

Entity Node	Derived Concrete Components				
	1st Generation	2nd Generation		3rd Generation	
		Alternatives	No. A	Alternatives	No. A
Head	Head 1	←	1	←	1
Pipe	Pipe 1	←	1	←	1
Hose	Hose 1	←	1	←	1
Dust pack	Bag 1	←	1	←	1
Turbine	Turbine 10	Turbines 2, 3, 10	3	Turbines 1–3, 9, 10	5
Code	Code 1	←	1	—	—
AC plug	AC plug 1	←	1	←	1
Filter	Filter 1	Filters 1, 6, 7	3	Filter 7	1
Main motor	Motor 1	Motors 1, 3, 10, 14–16, 18–2	9	Motors 1, 4, 14, 18	4
Head motor	Motor 21	←	1	Motors 21, 25–27	—
Body	Body 2	Bodies 3, 5, 6, 8, 10	5	Bodies 4–6	3
Converter	—	—	—	Converter 1	1
Inverter	—	—	—	Inverter 1	1
Variable resistance	V. resist. 1	←	1	←	1
Conductor	Lead wire 1	←	1	←	1
Substrate	Substrate 1	←	1	←	1
Roller	—	Roller 1	1	←	1
Winding mech.	Winding mech. 1	←	1	—	—
Battery	—	—	—	Battery	1
Charging body	—	—	—	Charg. body 1	1
Ion plate	—	—	—	Ion plate 1	1

(←) The parts selected in the previous generation continue to be used; (—) not applicable in this generation; No. A, number of alternatives.

components and, assuming the largest platform is adopted, they are divided into the platform (Table 6) and the upgrade modules (Table 7). As shown in Table 5, sets of components are selected to some entity node as alternatives, based on the “delayed selection of components” strategy. For example, for an entity “Turbine,” components “Turbine2,” “Tur-

bine3,” and “Turbine10” are listed in the second generation, and the designer should choose one among them until the upgrading to the second generation is actually executed. Figure 14 indicates the result of the enlarging and shrinking platform strategy. The largest and smallest boundaries indicated by the system were useful to determine the size of the platform. In this way, the prototype system succeeded in supporting the upgrade design based on the proposed methodology.

Fig. 14. Derived candidates of the platform.**Table 6.** Initial configuration in the largest platform

Platform	Modules	
	Module Name	Components
Motor 21	Module 1	Code 1, Winding mech. 1
Head 1	Module 2	Filter 1
Pipe 1	Module 3	Motor 1
Hose 1	Module 4	Body 2
V. resist. 1		
Substrate 1		
AC plug 1		
Dust pack 1		
Lead wire 1		
Turbine 10		

6. DISCUSSION

As described in Section 5, we succeeded in designing an upgradable vacuum cleaner based on the proposed methodology and the prototype system. Although the design result should be evaluated by experts, the proposed methodology, at least, systematically guided us in designing an upgradable product. The quality of the design result depends on the designer's ability and the sufficiency of related information. One of the main features of the proposed methodology is to guide a designer in designing upgradable products that can flexibly adapt to the uncertainty specified in a given upgrade plan by appropriately configuring platform and upgrade modules. This feature is verified by the case study because several candidates of platforms different in the size are indicated and sets of alternative components are derived. Therefore, we conclude that the design methodology proposed in this paper has feasibility and rationality.

As discussed in Section 3.1, the proposed methodology has some similarity to the existing various research works about the platform design. Certainly, most of them are quite sophisticated and mature. However, the main emphasis we wanted to address in this paper is to provide a design workspace that supports a designer to find candidates of functional changes in long-term usage of a single product, as opposed to a variety of different products, and candidates of configurations of a platform and modules by representing them explicitly. According to Ulrich's definition of the product architecture (1995), the proposed methodology succeeded in supporting arrangement of functional elements and mapping from functional elements to physical components in a flexible manner. The functional representation and the function design are obviously the most effective in solving these issues, while the representation might be inevitably too abstract.

In much of the literature, including Fujita et al. (1999), Li and Azarm (2002), Martin and Ishii (2002), and Messac et al. (2002), function structure is fixed and common among multiple products. However, function structure differs in each generation in this paper, and candidates of functional modification are given by the library of FBS fragments and function prototypes. In Fujita et al. (1999), without discussing changes of sizes of modules, alternatives of modules are given *a priori*. In Li and Azarm (2002), the process for finding out design alternatives is not supported. Although product structure is also given in Messac et al. (2002), this method gives a good suggestion to the parametric upgrading in this paper. Martin and Ishii (2002) also do not support the process of finding design modifications, and leaves this process to a designer by suggesting some rules. In this sense, their methodology is quite practical. Moreover, future requirements, which correspond to the upgrade plan in this paper, are not clarified but represented only as likelihood of future change. However, rules for design modification (viz., approaches reducing generational variety and coupling indexes) in their work have considerable similarity to our methodology. "1a. rearrange the mapping of functionality to components," of which process was not shown in their work, is the process we have solved in this paper. "2b. increase the 'headroom' of the specification" is the "making functions insensitive" principle in this paper, and we have shown how to realize this principle. Needless to say, these methodologies are indispensable for evaluating and optimizing design alternatives generated by the methodology proposed in this paper. However, without doubt, the process for ascertaining design alternatives should also be supported.

These features of our methodology are indispensable for supporting upgrade design in general. In addition, the way

Table 7. Upgrade modules

Module Name		Components
2nd Generation		
Additional modules	Module 5	Roller 1
Replacing modules	Module 4 → Module 6	(Bodies 3, 5, 6, 8, 19)
	Module 3 → Module 7	(Motors 1, 3, 10, 14–16, 18–20)
	Module 2 → Module 8	(Filters 1, 6, 7)
Removal modules	—	
3rd Generation		
Additional modules	Module 9	Charg. body 1, Ion plate 1
	Module 10	Converter 1, conductor 1, inverter 1, battery 1
Replacing modules	Module 6 → Module 11	(Bodies 4–6)
	Module 7 → Module 12	(Motors 1, 4, 14, 18)
	Module 8 → Module 13	Filter 7
Removal Modules	Module 1	

Among the set of components A, B, and C, a component should be selected as the delayed section of components.

to deploy the above-mentioned methodologies for the platform design toward upgrade design, which is an important issue in ecodesign, has not been shown well. Therefore, this design methodology may have impact on the ecodesign research community.

In contrast, upgrade design, as a general term, has not been practiced well. This issue mainly comes from two reasons: one is the future uncertainty we mentioned several times in this paper, and the other is the business strategy, where a company makes profits by providing upgrading services rather than just by selling products. Although this paper focuses on formalization of the design methodology for upgradability, these issues require development of business strategies, including upgrade planning, for upgrade design as an important future work. At least there should be some product categories suitable for upgrade design. Examples include facsimiles (Fujimoto et al., 2003), photocopiers, and printers. Conditions of such suitable products include the following:

- products that have short value lifetimes;
- products of which functions and their performance in some generations can be specified with some certainty; perhaps, it is desirable that their lifetimes are less than 5 years and, in this case, product development cycle, namely generation, might be shorter; and
- preferably, the company can control trends of products according to their upgrade plans. This might be possible in oligopolistic markets.

Although designing upgradable products is a challenging issue, it is worth challenging as a hopeful option of environmentally conscious life cycle design, because it will realize “servicification” of the manufacturing industry (Tomiyama, 1997) as well as mitigation of environmental issues. Using appropriately designed upgradable products for a long time will have higher environmental efficiency than replacing whole existing products. At the same time, they can provide higher additional value than the value provided just by replacing software. In fact, Nakamura and Kondoh (2001) clarified economic and environmental advantages of such products.

In contrast, because the objective of this paper focused on formalizing the upgrade design process, there are several issues to be solved, besides the above-mentioned issues of upgrade planning and business strategy:

- Because the FBS model symbolically represents the physical structure of a design object, it is difficult to deal with geometry of components and spatial constraints among them. This is one of the drawbacks to the symbolic representation of function in general. Although this kind of information is critical for determining configuration of modules, bridging symbolic functional representation with geometric information

is too large an issue in general to solve in this paper. This is one of our future issues.

- In the current implementation, the platform is determined progressively by considering each FBS fragment in each generation. Therefore, although the candidates of the platform derived in the case study are acceptable as far as our investigation, there might be a need for some extensions of the methodology for determining the platform from the viewpoint of overall generations.

Finally, let us discuss the impact of the proposed methodology to the functional representation and reasoning community. “Upgrade design” is a distinctive application of functional design because it inevitably requires representation and manipulation of functions. In contrast, in traditional design, the role of functions and advantages of functional representation and manipulation are often unclear to researchers outside of the community. Therefore, we believe that it is important for the community to excavate various applications to which functional representation and reasoning are indispensable, such as upgrade design. Various modeling methodologies for functions have been proposed in this community and FBS modeling is one of them. In terms of supporting the design methodology for upgradability, the required features of functional modeling are summarized as follows and FBS modeling satisfies them. Then, other functional modeling methodologies might also be available if they satisfy these conditions.

- To clearly represent a relationship between required functions and realizing structure is indispensable for determining a platform and upgrade modules. In FBS modeling, they are related via behavioral concepts. Although we could not make full use of functional and behavioral reasoning facility of FBS modeling and KIEF (Yoshioka et al., 2004), this facility is useful for discovering side effects of designer’s operations for adding upgradability and checking consistency of design results.
- To support a process to find physical embodiment from a required function is also indispensable. Although this feature is generally important in traditional design, FBS modeling has a high ability to support this process, constructing FBS fragments considering existing structure of the previous generation, in the design methodology for upgradability.

As an extension of FBS modeling, this paper discussed, at the step “parametric upgrade design” in Section 4, a method to relate abstract entity nodes in FBS modeling to concrete components through the range calculation and search through a component database. Because one of the drawbacks of symbolic representation of function is the difficulty to embody its abstract representation into con-

crete physical structure, this extension of FBS modeling provides a way to do this.

7. CONCLUSIONS

In this paper, we proposed a design methodology for upgradable products. Development of upgradable products is an important issue for solving environmental problems, especially reduction of resource consumption and waste amount, while extending business chances of manufacturing industries. Because future uncertainty is inevitable in planning the upgrading of a product, dealing with it is one of the main issues in upgrade design. The proposed methodology supports a designer to find an upgradable design solution that can flexibly adapt to the uncertainty specified as an upgrade plan by employing a functional modeling scheme, FBS modeling, and based on two strategies; delayed selection of components and enlarging and shrinking platform. The main features of the methodology include to support a designer for finding candidates of functional changes in a long-term usage of a single product and candidates of configurations of a platform and modules by representing these two kinds of candidates explicitly. In contrast, limitations include upgrade planning, evaluation of design results, and treatment of geometric constraints.

This paper mainly focused on formalization of the upgrade design process and modeling of upgrade design operations. The prototype system and a case study using the system verified feasibility of the proposed design methodology. However, this prototype system is developed just for verifying the methodology; we are planning to develop a more practical design support system by, for instance, introducing functional and behavioral reasoning facility embedded in the KIEF system.

Upgrade design is an application of functional design in which functional representation and manipulation are obviously indispensable. Therefore, it is important for the functional design community to develop such applications as the upgrade design.

Future works include the following:

- introducing information about value and physical lifetimes of components to decide platform and upgrade modules more practically;
- developing representation of interfaces between a platform and upgrade modules, which has not been discussed enough in this paper;
- introducing advanced modularization methodologies to decide upgrade modules more rationally; and
- developing a method for evaluating design results in terms of upgradability, flexibility against uncertainty, environmental consciousness, and profitability.

ACKNOWLEDGMENTS

A part of this work was financially supported by Inverse Manufacturing Forum, Japan, and Tokyo Metropolitan University. We

thank Y. Umemori, H. Yagi, Y. Ishigami, and A. Matsuda for their contributions to the theoretical development and implementation of the prototype system.

REFERENCES

- Bracewell, R.H., & Sharpe, J.E.E. (1996). Functional descriptions used in computer support for qualitative scheme generation—Schemebuilder. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10(4), 333–346.
- Chakrabarti, A., & Bligh, T.P. (2001). A scheme for functional reasoning in mechanical conceptual design. *Design Studies* 22(6), 493–517.
- Daimon, T., Kondoh, S., & Umeda, Y. (2004). Proposal of decision support method for life cycle strategy by estimating value and physical lifetimes. *Proc. 11th Int. CIRP Life Cycle Engineering Seminar*, pp. 49–56.
- Finch, W.W., & Ward, A.C. (1995). Generalized set-propagation operations over relations of more than three variables. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 9(3), 231–242.
- Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence* 24(3), 85–168.
- Fujimoto, J., Umeda, Y., Tamura, T., Tomiyama, T., & Kimura, F. (2003). Development of service-oriented products based on the inverse manufacturing concept. *Environmental Science & Technology* 37(23), 5398–5406.
- Fujita, K., Akagi, S., Yoneda, T., & Ishikawa, M. (1998). Simultaneous optimization of product family sharing system structure and configuration. *Proc. ASME 1998 Design Engineering Technical Conf.*, Paper No. DETC98/DFM-5722.
- Fujita, K., Sakaguchi, H., & Akagi, S. (1999). Product variety development and its optimization under modular architecture and module commonization. *Proc. ASME 1999 Design Engineering Technical Conf.*, Paper No. DETC99/DFM-8923.
- Fujita, K. (2002). Product variety optimization under modular architecture. *Computer-Aided Design* 34(12), 953–965.
- Gero, J.S., & Kannengiesser, U. (2004). The situated function-behaviour-structure framework. *Design Studies* 25(4), 373–391.
- Hiroshige, Y., Nishi, T., & Ohashi, T. (2001). Recyclability evaluation method (REM) and its application. *Proc. EcoDesign 2001*, pp. 315–320. Los Alamitos, CA: IEEE Computer Society.
- Ishii, K. (1999). Incorporating end-of-life strategy in product definition. *Proc. EcoDesign'99*, pp. 364–369. Los Alamitos, CA: IEEE Computer Society.
- Johnson, M.E., & Anderson, E. (2000). Postponement strategies for channel derivatives. *International Journal of Logistics Management* 11(1), 19–35.
- Jovane, F., & Alting, L. (1993). A key issue in product life cycle: disassembly. *Annals of CIRP'93* 42(2), 651–658.
- Kobayashi, H. (2001). Life cycle planning for strategic evolution of eco-products. *Proc. 12th Int. Conf. Engineering Design (ICED01)*, pp. 757–763.
- Li, H., & Azarm, S. (2002). An approach for product line design selection under uncertainty and competition. *Journal of Mechanical Design* 124, 385–392.
- Martin, M.V., & Ishii, K. (2002). Design for variety: Developing standardized and modularized product platform architectures. *Research in Engineering Design* 13(4), 213–235.
- Matsuda, A., Shimomura, Y., Kondoh, S., & Umeda, Y. (2003). Upgrade planning for upgradable product design. *Proc. EcoDesign 2003*, pp. 231–234. Los Alamitos, CA: IEEE Computer Society.
- Messac, A., Martinez, M. P., & Simpson, T. W. (2002). Introduction of a product family penalty function using physical programming. *Journal of Mechanical Design* 124, 164–172.
- Nakamura, S., & Kondoh, Y. (2001). Waste input-output analysis of disposal, recycling, and extended life of electric home appliances. *Proc. EcoDesign 2001*, pp. 814–819. Los Alamitos, CA: IEEE Computer Society.
- Prabhakar, S., & Goel, A.K. (1998). Functional modeling for enabling adaptive design of devices for new environments. *Artificial Intelligence in Engineering* 12(4), 417–444.
- Qian, L., & Gero, J.S. (1996). Function-behavior-structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10(4), 289–312.

- Sembugamoorthy, V., & Chandrasekaran, B. (1986). Functional representation of devices and compilation of diagnostic problem-solving systems. In *Experience, Memory, and Reasoning* (Riesbeck, C.K., Ed.). Hillsdale, NJ: Erlbaum.
- Simpson, T.W., Maier, J.R.A., & Mistree, F. (2001). Product platform design: method and application. *Research in Engineering Design* 13(1), 2–22.
- Suh, N.P. (1990). *The Principles of Design*. New York: Oxford University Press.
- Suwa, M., Gero, J.S., & Purcell, T. (1999). Unexpected discoveries and s-inventions of design requirements: a key to creative designs. In *Computational Models of Creative Design IV* (Gero, J.S., & Maher, M.L., Eds.), pp. 297–320. Sydney, Australia: Key Centre of Design Computing and Cognition, University of Sydney.
- Tomiyama, T. (1997). A manufacturing paradigm toward the 21st century. *Integrated Computer Aided Engineering* 4, 159–178.
- 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. New York: Wiley.
- Ulrich, K.T. (1995). The role of product architecture in the manufacturing firm. *Research Policy* 24, 419–440.
- Umeda, Y., Hijihara, K., Oono, M., Ogawa, Y., Kobayashi, H., Hattori, M., Masui, K., & Fukano, A. (2003). Proposal of life cycle design support method using disposal cause analysis matrix. *Proc. 14th Int. Conf. Engineering Design (ICED03)*.
- Umeda, Y., Ishii, M., Yoshioka, M., & Tomiyama, T. (1996). Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10(4), 275–288.
- Umeda, Y., Nonomura, A., & Tomiyama, T. (2000). Study on life-cycle design for the post mass production paradigm. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 14(2), 149–161.
- Umeda, Y., Shimomura, Y., Yoshioka, M., & Tomiyama, T. (1999). A proposal of design methodology for upgradable products. *Proc. ASME 1999 Design Engineering Technical Conf.*, Paper No. DETC99/DFM-8969.
- 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. Southampton/Berlin: Computational Mechanics Publications/Springer-Verlag.
- Umeda, Y., & Tomiyama, T. (1997). Functional reasoning in design. *IEEE Expert: Intelligent Systems and Their Applications* 12(2), 42–48.
- Umemori, Y., Kondoh, S., Umeda, Y., Shimomura, Y., & Yoshioka, M. (2001). Design for upgradable products considering future uncertainty. *Proc. EcoDesign 2001*, pp. 87–92. Los Alamitos, CA: IEEE Computer Society.
- Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., & Tomiyama, T. (2004). Physical concept ontology for the knowledge intensive engineering framework. *Advanced Engineering Informatics* 18, 95–113.

Yasushi Umeda has been a Professor in the Department of Mechanical Engineering, Graduate School of Engineering,

Osaka University, since 2005. Prior to this appointment, he was an Associate Professor at Tokyo Metropolitan University until 2005 and a Lecturer at the University of Tokyo until 1999. He received his PhD in precision machinery engineering from the Graduate School of the University of Tokyo in 1992. Dr. Umeda's research interest includes design theory and methodology, environmental conscious design, life-cycle engineering, life-cycle simulation, maintenance engineering, and knowledge intensive engineering.

Shinsuke Kondoh has been a Researcher at the National Institute of Advanced Industrial Science and Technology since 2005. He received his PhD in precision machinery engineering from the Graduate School of the University of Tokyo in 1999. Dr. Kondoh's research interest includes life-cycle engineering (ecodesign and life-cycle simulation), manufacturing system design, self-organization, and distributed autonomous system (cellular machines).

Yoshiki Shimomura has been a Professor in human mechatronics systems in the Faculty of System Design, Tokyo Metropolitan University, since 2005. Prior to this appointment, he was an Associate Professor in research into artifacts at the Center for Engineering, University of Tokyo. He received his PhD in precision machinery engineering from the Graduate School of the University of Tokyo in 1997. Dr. Shimomura's research interest includes service engineering, life-cycle engineering, design theory and methodology, intelligent systems, reasoning mechanism, and soft machines (self-maintenance machines and cellular machines).

Tetsuo Tomiyama is a Professor of intelligent mechanical systems in the Faculty of Mechanical, Maritime, and Materials Engineering at the Delft University of Technology. Prior to this appointment in 2002, he was a Professor in research into artifacts, Center for Engineering, University of Tokyo. He received his PhD in precision machinery engineering from the Graduate School of the University of Tokyo in 1985. Dr. Tomiyama's research interest includes design theory and methodology, knowledge intensive engineering, service engineering, applications of qualitative physics, large-scale engineering knowledge bases, soft machines (self-maintenance machines and cellular machines), and life-cycle engineering.