Proceedings of IDETC/CIE 2006
ASME 2006 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
Philadelphia, Pennsylvania USA, September 10 – 13, 2006

DETC2006-99360

KNOWLEDGE MODEL FOR MANAGING PRODUCT VARIETY AND ITS REFLECTIVE DESIGN PROCESS

Yutaka Nomaguchi, Tomohiro Taguchi and Kikuo Fujita
Department of Mechanical Engineering
Osaka University
Suita, Osaka 565-0871, JAPAN
Email: noma@mech.eng.osaka-u.ac.jp
Tel: +81-6-6879-7324, Fax: +81-6-6879-7325

ABSTRACT

Recent manufacturers have been utilizing product families to diversify and enhance the product performance by simultaneously designing multiple products under commonalization and standardization. Design information of product architecture and family is inevitably more complicated and numerous than that of a single product. Thus, more sophisticated computer-based support system is required for product architecture and family design. This paper proposes a knowledge model for a computer-based system to support reflective process of designing product architecture and product family. This research focuses on three problems which should be overcome when product family are modeled in the computer system; design repository without data redundancy and incorrectness, knowledge acquisition without forcing the additional effort on the designer, and integration of prescriptive models to support early stages of the design process. An ontology that is a foundation of a knowledge model is defined to resolve these problems. An example of designing an air conditioner product family is shown to demonstrate the capability of the system.

Keywords: Product architecture, product family, knowledge model, knowledge acquisition, design rationale, design support, ontology

1 INTRODUCTION

As manufactures adapt to recent highly competitive global marketplace, need of integrating various high-end performances and cost reduction of product increases. They are utilizing product families to diversify and enhance the product performance by simultaneously designing multiple products under commonalization and standardization (Simpson *et al.*, 2006). The key to rational and successful product family design is product architecture (Ulrich, 1995). Platform design of automobiles, i.e., Volkswagen's A-Platform (Wilhelm, 1997), is a representative example, which shows the power of established product architecture.

This research defines product architecture as the schema which connotes each product variant by defining possible structures and possible attribute values. Product architecture is generally written by multiple aspects, such as customer needs, functions and entities. Design information of product architecture and family is inevitably more complicated and numerous than that of a single product. Thus, a more sophisticated computer-based support system is required for product architecture and family design. When product family is modeled in a computer system, a major problem is product description of multiple aspects on a product family without data redundancy and incorrectness even in large-scaled design repository (McKay *et al.*, 1996). Another problem is that supporting iterative process, in which a designer performs trials

and errors reflectively, is inevitable in an early stage of the design process. Even though many research groups have been tackling computational methods to optimize product architecture and family design (Simpson, 2004) in the later stages, the conceptual design stage should be supported by a prescriptive methodology such as QFD and GVI/CI (Martin and Ishii, 2002). It is valuable to externalize his/her design knowledge and then to reflectively consider it.

This paper proposes a knowledge model of product architecture and family which solves the above mentioned problems. For design repository without data redundancy and incorrectness, this research defines an ontology of product architecture and family design. For supporting reflective design process, a knowledge management oriented design support system which we have been developing (Nomaguchi *et al.*, 2004) is used. This system automatically captures design process and generate an argumentation structure based on IBIS (Issue Based Information System) (Kunz and Rittel, 1970) in order to represent a designer's reflective process. To employ this system, a knowledge model should have the following two functions; (i) to formally capture design process, an ontology that is a foundation of a knowledge model should define not only concepts of a result of design but also concepts of any state of design and (ii) to integrate prescriptive design models, model-dependent concepts should be defined as an extended part of the ontology.

The following part of this paper consists of six sections. Section 2 describes our perspective of knowledge. Section 3 proposes an ontology of product architecture and family design. Section 4 briefly explains the features of a knowledge management oriented design support system which we have been developing (Nomaguchi *et al.*, 2004). Section 5 illustrates implementation and a design example. In Section 6, characteristics of our research are discussed by comparing related works. Section 7 concludes this paper.

2 KNOWLEDGE MODEL

As knowledge management has become a new hot topic in the last decade, knowledge modeling is one of key issues of design research (e.g., Rosenman and Simoff, 2001). Different researchers defined knowledge to investigating questions; 'what is knowledge?' and 'how is it produced?' This section describes our perspective of knowledge.

2.1 Reflective Design Process and Knowledge

In general, human's problem solving including design is done through an iteration process (Simon, 1969). Knowledge takes a crucial role to do effective trials in an iteration process as much as possible and to aim at a goal. Schön (1982) analyzed knowledge of professionals such as medical doctors, architect designers and engineers, and proposed two models of knowledge. One is knowledge of a technical expert, which is systematized and can be applied to practices in an explicit manner. The other is knowledge of a reflective practitioner, which is not systematized and should be captured through reflective process of practices. In design process, some of knowledge is based on an already-systematized theory such as physics. Besides, a designer acquires many of knowledge through experiences of iterations. As Schön stated that the latter knowledge is essential to tackle recent complex problems, acquisition of it is required for knowledge management.

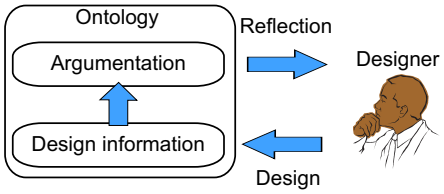

Figure 1. Ontology based reflective design process

To formally capture reflective design process, this research proposes an ontology based approach. An ontology of design works as a foundation of reflective design process as shown in Figure 1. Firstly, a designer creates design information on an ontology. Secondly, an argumentation structure in design process is elicited based on an ontology. Finally, a designer reflects his/her design process by elicited argumentation structure. Based on this reflection, a further design is done. While Schön focuses on a certain pattern of reflection in mind, we focus on formal elicitation of reflection. A framework discussed in this research supports further reflection in mind by formally capturing argumentation structure of design process which is done as a result of reflection.

2.2 Knowledge Management Oriented Design Support

We have been researching a knowledge management oriented design support framework to dynamically acquire designer's knowledge, which is used in reflective design process, as a by-product of design without forcing additional effort on a designer (Nomaguchi *et al.*, 2004). It is an implementation of an ontology based approach to formally capture reflective design process.

This framework formally captures design process at three levels as shown in Figure 2; log of designer's actions, state transition of design information and argumentation structure of a designer's thinking process. The third level represents reflective process explicitly, and it is composed automatically based on the first level. In this framework, a knowledge model, which is based on an ontology of design, takes an important

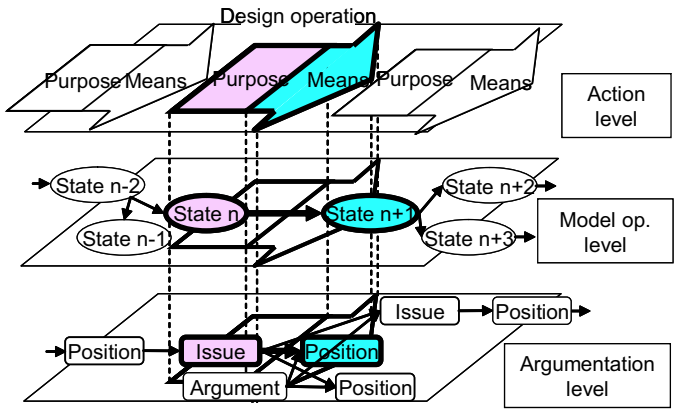

Figure 2. Design process model integrating Action, Model operation and Argumentation

role. A log of designer's actions is captured as a sequence of operations to a knowledge model. A state transition of design information is captured as a state transition of a knowledge model. An argumentation structure of a designer's thinking process is captured as a structure of purpose and means of an operation to a knowledge model.

From the viewpoint of this framework, a knowledge model should meet the following requirements. Firstly, a knowledge model should record not only the result of design but also states of design at a certain moment of design process. Secondly, a knowledge model should integrate various design models to represent design information from various aspects such as customer needs, functionality, structure and cost. Lastly, operations to a knowledge model should be defined by considering its purpose and means, which corresponds to an issue and a position of argumentation, respectively.

2.3 Toward Product Architecture and Product Family Design

Process of product architecture and family design also includes iteration process to reflectively consider problems such as 'which product should be in a family?,' 'how product architecture should be?,' 'which component should be commonalized?' and so on. To support reflective design process, some prescriptive methods have been developed, e.g. the market segmentation grid (Meyer, 1997) which is to plan family deployment strategy, GVI/CI (Martin and Ishii, 2002) which is QFD based indices to evaluate product architecture from the viewpoint of robustness against market changes. It is valuable for a designer by using such prescriptive models to externalize his/her knowledge, then to consider various ideas reflectively.

In order to apply the knowledge management oriented design support system to design of product architecture and product family, this paper defines a ontology that is a foundation of a knowledge model of product architecture and product family. This paper mainly discusses the first and the second requirements stated in Subsection 2.2. The last requirement is remained as our future work.

3 MODELING PRODUCT ARCHITECTURE AND PRODUCT FAMILY

3.1 Overview of Modeling Ontology

This section explains modeling concepts of a knowledge model, which is proposed for product architecture and family design. An ontology is a basis of knowledge modeling. Note that we interpret ontology in the sense; a theory about the sorts of concepts, properties of concepts, and relations among concepts that are defined in a specified domain of knowledge.

An ontology proposed in this research consists of two layers; a model-independent layer and a model-dependent layer. A model-independent layer consists of concepts to represent product architecture and product family which is independent from any specific prescriptive design model. A model-dependent layer consists of concepts to represent specific prescriptive design models. A concept of the latter layer is defined as a subclass of model-independent layer concepts. The combination of the above two layers makes expansibility higher. When a new prescriptive design model is integrated, new concepts is added only to model-dependent layer while model-independent layer is not modified.

3.2 Illustrative Example of Product Architecture and Family Deployment

This subsection introduces a design example used throughout this paper in advance; a design of indoor unit family of an air conditioner. Figure 3 shows a typical indoor unit of an air conditioner. The product architecture of entity aspect are shown in Figure 4 by UML (Unified Modeling Language). A rectangle

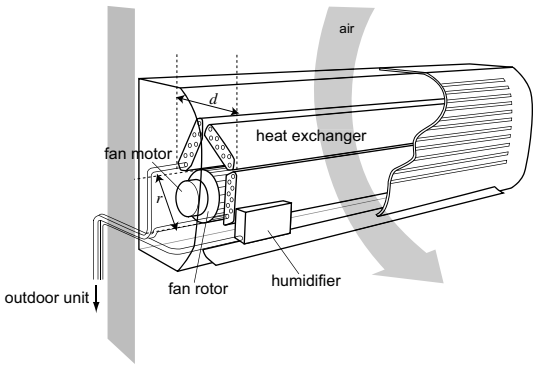

Figure 3. Schematic of indoor unit of air conditioner

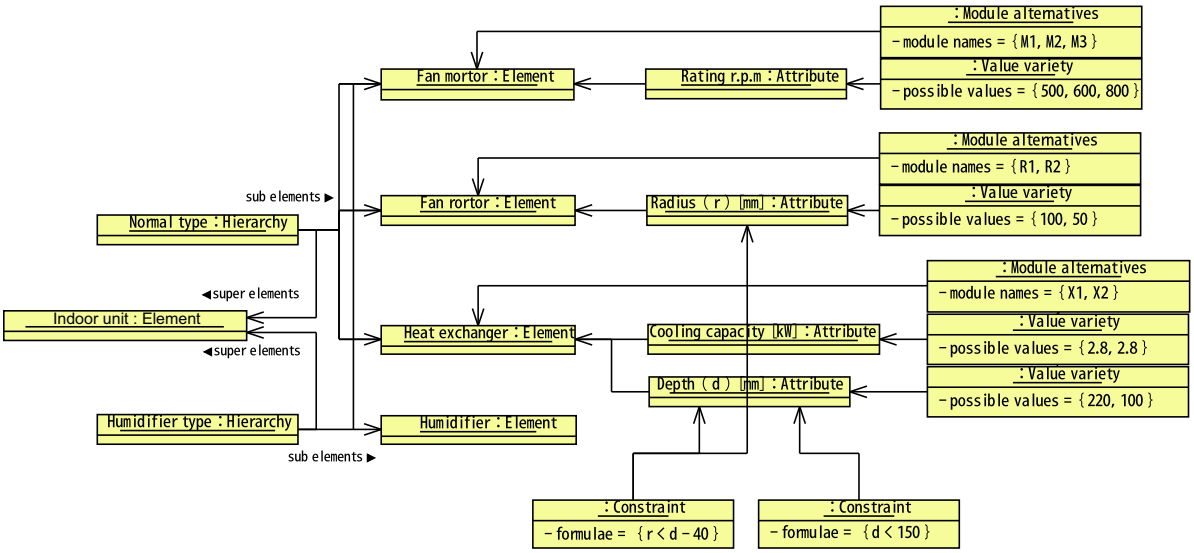

Figure 4. Product architecture of entity aspect of air conditioner indoor unit

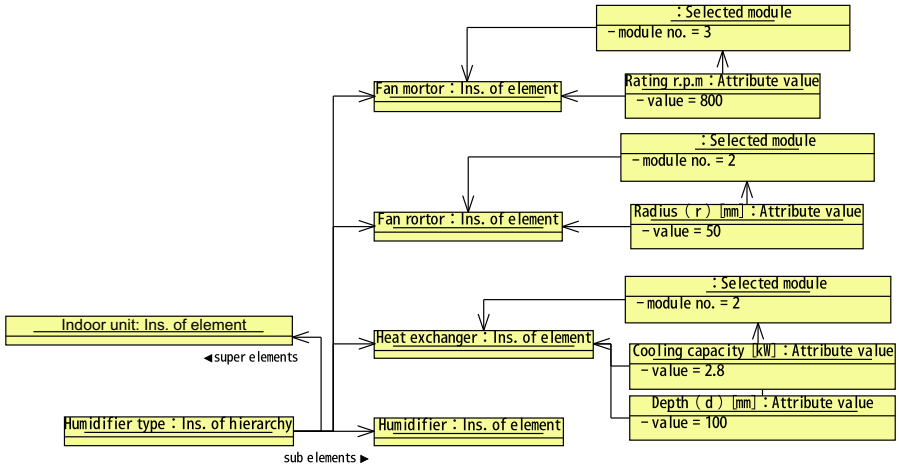

Figure 5. Indoor unit variant

node means an instance of a defined concept. Figure 4 shows a part of entity aspect structure, but it would be enough to understand the overview of a proposed knowledge model.

In Figure 4, there are two possible hierarchical structures of an indoor unit. One is a normal type hierarchy, which has a fan motor, a fan rotor and a heat exchanger as a subcomponent of an indoor unit. The other is an humidifier type hierarchy, which has a humidifier in addition to the normal type structure. A part of attributes which are used to specify an indoor unit variant are also shown in Figure 4; rating round per minute of a fan rotor, radius (r) of a fan rotor, cooling capacity of a heat exchanger and depth (d) of a heat exchanger. A value of the attribute of the component can be selected from the possible values defined by the module alternatives.

The combination of possible hierarchy types and possible modules results in 24 possible variants of the indoor unit family. However, constraints reduce the potential variety. In this example, there are two constraints; a constraint on depth of a heat exchanger '$d < 150$' and a constraint on radius of a fan rotor '$r < d - 40$.' Because of these constraints, 6 variants are possible. Figure 5 shows one of the possible variants.

3.3 Model-independent Concept

This subsection introduces concepts of a model-independent layer. Concepts in this layer form a taxonomical hierarchy as

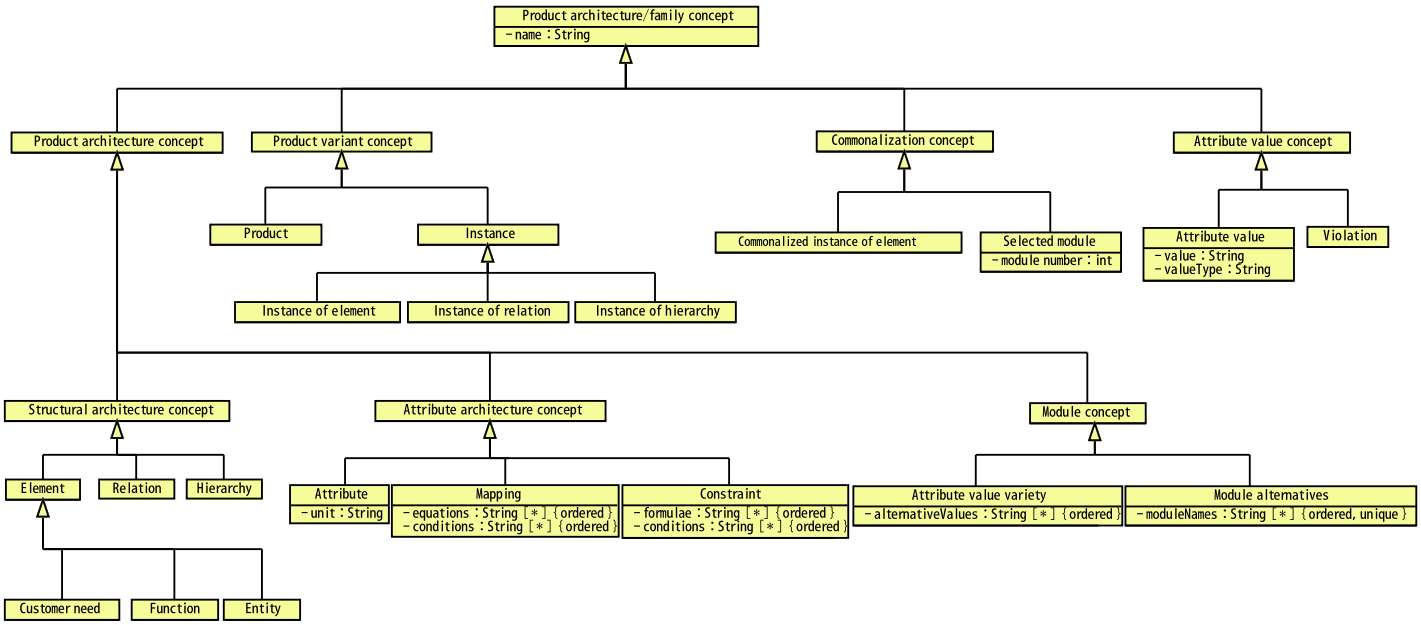

Figure 6. Taxonomy of modeling concepts

shown in Figure 6. The concept taxonomy, properties of concepts and associations between concepts are represented in the UML format.

A basic idea to represent product family in this research is as follows. Firstly, product architecture is represented as a schema which connotes the all possible structures and the possible value of the attribute. A product variant is represented as an instance of the product architecture. Then, commonalization and standardization is defined among product variants. Finally, detailed attribute values which characterize each product variant are defined. Therefore, the modeling concepts of this layer can be categorized into four; product architecture concept, product variants concept, commonalization concept and attribute value concept.

3.3.1 Product architecture concept Product architecture concept has the following sub concepts; structural architecture, attribute architecture and module. Structural architecture is a concept representing structural view point of product architecture. This has the following sub concepts. Figure 7 shows associations defined among the structural architecture concepts.

Element is a concept that constructs a product architecture. This is further categorized into three concepts; customer need, function and entity. For example, an element of customer need of an air conditioner is 'comfortable temperature,' 'save energy,' etc., an element of function is 'to control air temperature,' etc., and an element of entity is 'fan rotor,' 'heat exchanger,' etc.

Hierarchy is a concept that represents a possible hierarchical relationship between elements. A hierarchy node has an association to an element, which is super level of the hierarchy, and elements, which are sub level of the hierarchy. In the example, a hierarchy node 'humidifier type' represents that 'indoor unit' has four sub entities; 'heat exchanger,' 'fan rotor,' 'fan motor' and 'humidifier.'

Relation is a concept that represents a relation between elements. For example, 'Function-Structure relation' is defined among a function 'to control air temperature' and entities 'fan motor,' 'fan rotor,' 'heat exchanger.'

Attribute architecture is a concept representing a mathematic constraint which reduces possible product variants. This has following three sub concepts. Figure 8 shows associations defined among attribute architecture concepts.

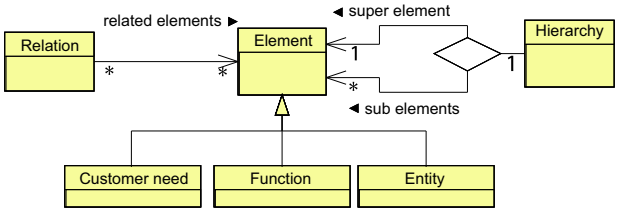

Figure 7. Associations of structural architecture concepts

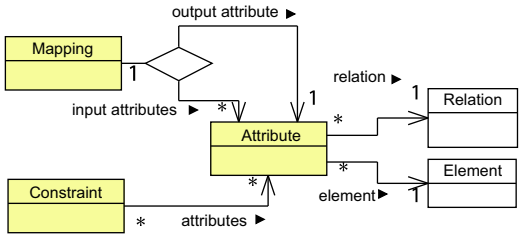

Figure 8. Associations of attribute architecture concepts

Attribute is a concept that represents a character of an element or a relation. An attribute node has an association to an element or a relation. In the example of Figure 4, two attributes of entity 'heat exchanger' are defined; 'cooling capacity' and 'depth.' An attribute node has a unique attribute value.

Mapping is a concept that represents existence of a numerical function which determines a value of an attribute. A mapping node has an association to one output attribute and multiple input attributes. A list of pairs of an equation and its condition is defined as a property of a mapping node.

Constraint is a concept that represents existence of a constraint among attribute values. A constraint node has an association to multiple attributes. A list of pairs of a formula and its condition are defined as a property of a constraint node. In Figure 4, two inequality constraints '$r < d - 40, d < 150$' are defined for a heat exchanger's depth (d) and a fan rotor radius (r).

Module is a concept representing available modules of an entity. This has the following two sub concepts. Figure 9 shows associations defined among module concepts.

Module alternative is a concept which represents existence of available modules for an entity. A module alternative node has an association to one entity node. A list of module names is defined as property of a module alternative node. In Figure 4, a heat exchanger has two module alternatives; X1 and X2.

Attribute value variety is a concept which represents attribute values of each module which is defined by a module alternative node. A value variety node has an association to one attribute node and one module alternative node. A list of possible values is defined as property of an attribute value variety node. Possible values are defined in the same order as module names of an associated module alternative node. In Figure 4, X1, a module of a heat exchanger, has the following attribute values; cooling capacity $= 2.8$, depth $=$ 220.

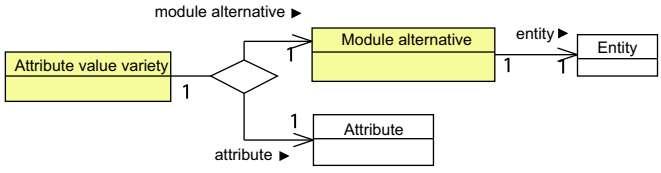

Figure 9. Associations of module concepts

3.3.2 Product variant concept

Product variant concept represents a structure of a product in a product family. Figure 10 shows associations defined among product variant concepts. This concept has the following two sub concepts.

Product is a concept which represents a product variant in a product family.

Because product architecture connotes all possible structures, a structure of a product variant is represented as an instance of concepts of product architecture. To represent a product variant in this way, this research defines *instance concept*. Note that *instance* is used as the similar sense of object-oriented programming; a substance of information of a class that defines properties and methods of an object. In this definition, structural architecture concepts correspond to a class of instances.

Instance is a concept which represents that a product has an instance of product architecture concept. Three sub concepts corresponding to structural architecture concepts are defined; instance of element, instance of relation and instance of hierarchy. Each of an instance node has an association to a product node and a 'class' concept of product architecture.

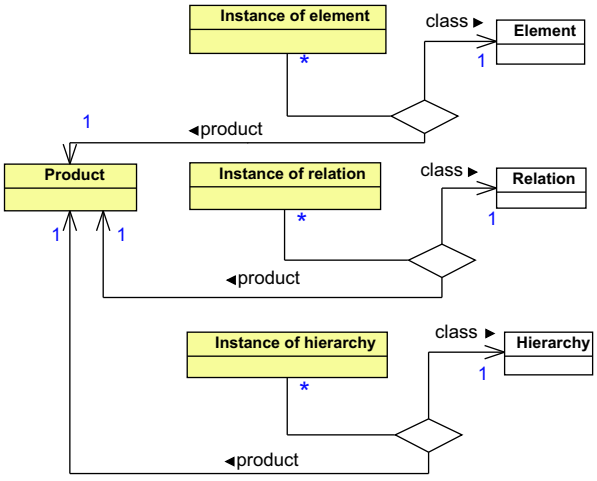

Figure 10. Associations of product variant concepts

An instance node means that a product inherits a ‘class’ concept of product architecture, and that a product also inherits attributes, mappings, constraints, module alternatives and attribute value varieties which are associated to a class concept. This mechanism serves to reduce the amount of information required to represent product family.

3.3.3 Commonalization concept Commonalization concept represents commonality among product variants. The following two sub concepts are defined. Figure 11 shows associations related to commonalization concepts.

Commonalized instance of element is a concept which represents intention of a designer to commonalize components or parts of multiple product variants. A node of this concept has an association to multiple nodes of *instance of element*.

Selected module is a concept which represents a module selected for an instance of entity. A selected module node has a module id number as property, which designates a selected module name and a selected possible value. A selected module node has an association to a module alternative and an instance of element.

3.3.4 Attribute value concept Attribute value concept has two sub concepts; attribute value and violation. Figure 12 shows associations related to attribute value concepts.

Attribute value is a concept which represents a value of an attribute of an instance node. An attribute value node has a value and a value type as property.

There are four association types for an attribute value node as shown in Figure 12. A different association means a different way of determining an attribute value.

Association to an attribute and an instance represents that an attribute value of an instance is determined. An instance node in this definition can be omitted when it is a default value for all product variants.

Association to an attribute, an instance, a mapping and attribute values represents that an attribute value is determined as a result of calculating a mapping equation with input attribute values. An instance node in this definition can be omitted when it is a default value for all product variants.

Association to an attribute and a commonalized instance of element represents that an attribute value is determined as a result of commonalization.

Association to an attribute, a value variety and a selected module represents that an attribute value is determined as a result of selecting a module.

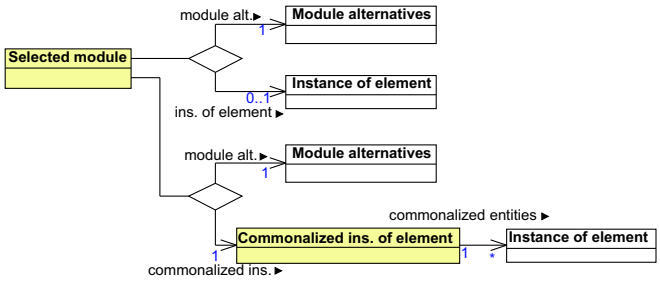

Figure 11. Associations of commonalization concepts

After an attribute value is determined, it should be checked whether it satisfies a constraint or not. If not, violation node is created to indicate a designer.

Violation is a concept which represents an attribute value violates a constraint. A violation node has an association to attribute values and a constraint.

Violation should be resolved by altering input attribute values or by relaxing a constraint. Note that the knowledge model

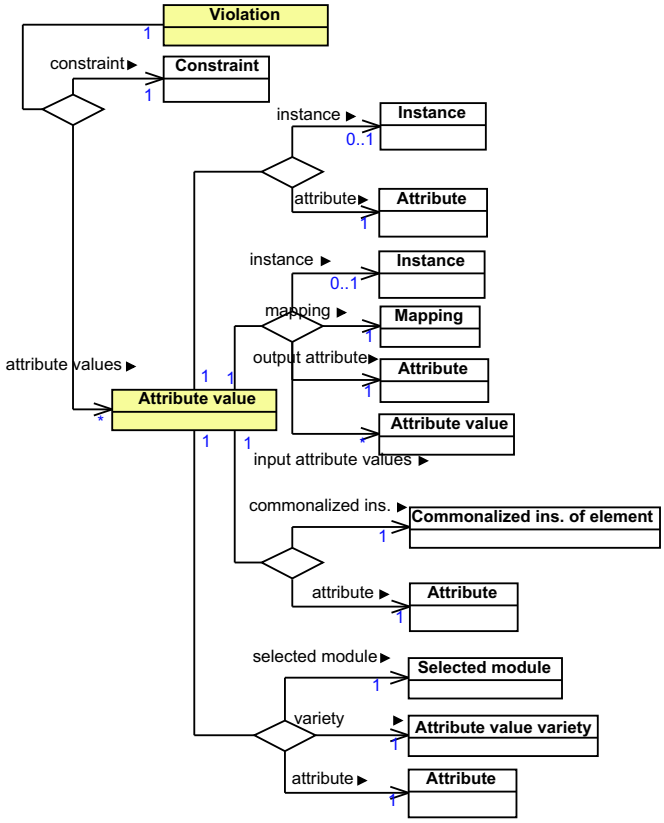

Figure 12. Associations of attribute value concepts

does not take charge of resolving constraint violation. The knowledge model does just describe a state of design which includes constraint violation.

3.4 Model-dependent concept

This subsection introduces prescriptive models which this research employs. A concept necessary to express these models is defined as a subclass of a model-independent concept. There is a possibility that other concepts in addition to concepts explained in this subsection are defined if necessary when the other model is integrated to the knowledge model.

3.4.1 Value graph, Function-Structure mapping

Value graph describes development of a customer need 'good product' into sublevel customer needs (see Figure 17-①). Function-structure mapping describes development of functions and components of a product, and relationships between functions and components (see Figure 17-③). Four concepts; customer need, function, entity and hierarchy are used to represent a development hierarchy.

3.4.2 Market segmentation grid

Market segmentation grid is a chart to segment a market (Meyer, 1997) (see Figure 17-②). Two axes, a segment (horizontal) axis and a scale (vertical) axis, and grids of an axis divides a market into niches. By allocating a product variant to a niche, a designer can discuss a strategy of product variant deployment. This research uses a customer need concept to represent an axis of an market segment grid. In addition, the following concepts are introduced.

Grid name represents grid of an axis. This is defined as a name of a utility function of *function-customer need mapping* which is introduced in subsubsection 3.4.3. For example, $20m^2$ or $28m^2$ is a grid name of 'appropriate room size' (see Figure 17-②).

Target is a sub class concept of attribute. This represents a grid name which a product variant is allocated as a target niche. For example, a product variant C28 aims for a target '$20m^2$' of 'appropriate room size' and for a target 'very thin' for 'good looking.'

3.4.3 Utility function model

Utility function model is used to evaluate customer's satisfaction by utility functions and attributes (Nomaguchi and Fujita, 2005). The following additional concepts are introduced to represent the utility function model.

Satisfaction degree is a sub class concept of attribute. This represents a degree of satisfaction of a customer need by a value of 0 to 1.0.

Function-customer need mapping is a sub class concept of mapping. This represents a set of utility functions which calculate a satisfaction degree of a customer need by a value of functional attribute. A utility function is represented by a quadratic function or an exponential function. For example, a satisfaction degree of 'appropriate room size' is calculated by a value of 'cooling/warning capacity' and utility functions shown in Figure 17-④.

Entity-function mapping is a sub class concept of mapping. This represents an equation to calculate a value of functional attribute. For example, a functional attribute 'COP (index of saving energy capacity)' is calculated by an equation P_o/w, where P_o is a capacity of an outdoor unit and w is a watt consumption.

3.4.4 QFD

QFD(Quality Function Deployment) describes correlation numbers between customer needs and functions, and ones between functions and components. These correlation numbers are used to deploy weights of customer needs to weights of components by simple matrix calculation. The following additional concepts are introduced to represent QFD.

C-F relation and F-S relation are both sub class concept of relation. They are used to represent the existence of correlation between a customer need and a function, and one between a function and a component.

Weight is a sub class concept of attribute. It is used to represent a weight of a customer need, a function and a component.

QFD correlation is a sub class concept of attribute. It is used to represent the correlation number of a C-F relation or an F-S relation.

3.4.5 Cost/Worth Graph

Cost/worth graph describes a balance of relative worth between relative cost of a component. The following additional concepts are defined to represent cost/worth graph.

Relative worth is a sub class of attribute. This is calculated by regularization of weights of components which are calculated by QFD.

Cost is a sub class concept of attribute. It is used to represent cost of a component.

Relative cost is a sub class concept of attribute. It is used to represent relative cost of a component. Its value is calculated by regularization of cost of components.

4 KNOWLEDGE MANAGEMENT ORIENTED DESIGN SUPPORT SYSTEM

This research employs a knowledge management oriented design support system which we have been developing (Nomaguchi *et al.*, 2004) in order to support reflective process of

designing product architecture and product family. This section explains the features of the system to formally capture reflective design process.

4.1 Data Consistency Management based on JTMS

A TMS (Truth Maintenance System) has been used in artificial intelligence research groups in order to cache for all inferences ever made (e.g., Doyle, 1979). This research employs a simple justification-based TMS mechanism to capture all design states and to track each of them anytime without redundancy and incorrectness. Any state of designing product architecture and product family is described as a network graph of concept nodes which are defined by Section 3. Each concept node is recorded in *In* node list, which keeps active concept nodes, or *Out* node list, which keeps inactive concept nodes, at a certain design state. When a new concept node n_n is going to be added, a JTMS mechanism maintains *In*/*Out* of nodes by following procedure;

(i) Searching *In*/*Out* node list to find a concept node n_s which is the same as n_n.
(ii) If n_s is found, adding n_s to *In* node list and $n_n := n_s$.
(iii) Adding a justification to n_n from nodes which are associated with n_n.
(iv) Collecting nodes $Con(n_n)$, which are contradictory to n_n.
(v) Adding all nodes in $Con(n_n)$ to *Out* node list.
(vi) Adding all nodes, which are justified only by a node in $Con(n_n)$, to *Out* node list.

Here, a *same* node is defined as follows; a node which has the same properties and the same associated nodes. A *contradictory* node is defined as follows; a node which violates a definition of association. However, there are some exceptions. For attribute value concept, for example, a contradictory node is defined as follows; a node which has the same properties, and has an association to the same attribute and the same instance.

4.2 Capturing Argumentation by Design Operation

As stated in Subsection 2.2, this system automatically generates an argumentation description to externalize reflective design process by a log of design operations which a designer has performed. Figure 2 illustrates the relationship between a log of design operations and argumentation model.

4.2.1 Argumentation model

This research employs IBIS (Kunz and Rittel, 1970) based argumentation model. IBIS represents argument structure that includes issues and their positions by a network graph. Figure 13 depicts IBIS model. The model consists of the three types of node. *Issue* is a node, which represents an issue discussed in argument. *Position* is a node that represents one of multiple alternative solutions to an issue. A

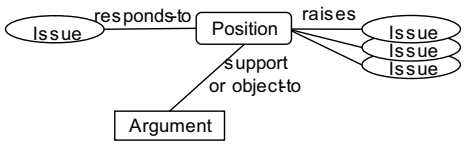

Figure 13. Argument model — IBIS —

position node has a relation *respond-to* to an issue, to which a position gives a solution. A position node can be followed by an issue node, when a new issue is raised from a position. In this case, a relation *raise* is defined between a position node and an issue node. *Argument* is a node that represents an argument among multiple positions. If an argument supports a position, a relation *support* is defined between them. If an argument objects to a position, a relation *objected-to* is defined.

Each node has the following properties.

Text description : a description that explains contents of an IBIS node.
Operation type : a name of design operation by which an IBIS node is generated.
Focused node list : a list of concept nodes in a product architecture/family model that is argued in an IBIS node.
Design state list : a list of design states in which an IBIS node is generated.

4.2.2 Generating argumentation structure

A design operation takes an important role for generating an argumentation structure. A design operation is defined; an operation on a design object model which explicitly defines its purpose and means. This research defines 29 design operations for product architecture/family design process. Figure 14 shows defined design operations. Figure 15 shows a definition of *develop function of product architecture*. A design operation defines *operation primitives*, which are operations to JTMS as explained in Subsection 4.1. When a design operation is performed, operation primitives of the list are performed.

A design operation defines*referred concept nodes*, which are requisites of an operation, and *added concept nodes*, which are added as a result of an operation. In Figure 15, a function node is referred and multiple function nodes are added as a result of this operation.

Under the above definition of design operations, the following procedure generates IBIS description after a design operation O_n is performed;

(i) Creating a new issue node I_n, which has referred concept nodes of O_n in its focused node list, and has operation name of O_n in its operation type.
(ii) Searching an issue node I_s which is the same as I_n. If I_s is found, $I_n := I_s$.

Prescriptive model	Product architecture	Product variant
-	Set product architecture name	Set product names
Value graph	Set toplevel customer needs Develop customer needs	Set toplevel customer needs Develop customer needs
Market Segment Grid	Set grid for customer needs	Set target segment
Function-Structure Mapping	Set toplevel function Develop function Set toplevel entity Develop entity	Set toplevel function Develop function Set toplevel entity Develop entity
QFD	Set correlation between customer needs and functions Set correlation between function and entities	Set weight of customer needs
Cost/Worth graph		Set cost of entity
Utility function	Set module alternatives Set commonalization Set attribute of entity Set attribute of function Set entity-function mapping Set function-customer needs mapping Set constraint	Set attribute value of entity Select module

Figure 14. Design operation

Operation name:	Develop function of product architecture
Refered nodes:	function : Function
Added nodes:	subFunctions : Function [*]
Purpose:	To decompose function to sub functions.
Means:	To add subFunctions as sub functions of function.
Operation primitives:	adding Function nodes, adding Hierarchy node

Figure 15. Design operation example

(iii) Creating a new position node P_n, which has added concept nodes of O_n in its focused node list, and has operation name of O_n in its operation type.
(iv) Searching a position node P_s which is same as P_n. If P_s is found, $P_n := P_s$.
(v) Searching a position node P_{n-1}, which focused node list contains at least one of focused nodes of I_n.
(vi) Connecting *raised* relation from P_{n-1} to I_n.
(vii) Connecting *respond-to* relation from I_n to P_n.

Here, a *same* IBIS node is defined as follows; a node which has the same operation type and the same focused node list.

5 IMPLEMENTATION

5.1 System Architecture

Based on the discussion above, a knowledge management oriented support system for product architecture and family

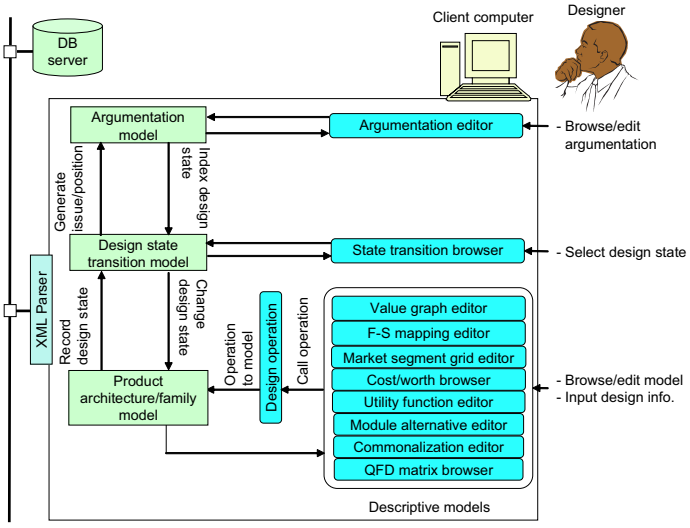

Figure 16. System architecture

design was developed in Java programming language (jdk 1.4.1) on Windows XP. Figure 16 shows the architecture of the system.

Design process on the system is carried out by using prescriptive models; value graph, function-structure mapping, market segmentation grid, cost/worth graph, utility function model, QFD, module alternative browser and commonalization browser. The system calls design operations to the knowledge model when a designer input design information on the prescriptive models. Design process performed by a designer is automatically recorded in three levels; action level, model operation level and argumentation level. A designer can edit description of an argument node. A recorded design process is stored in database in XML format.

5.2 Design Example

This subsection illustrates an application to designing family of an indoor unit of an air conditioner for demonstrating the capabilities of the implemented system. This design example is based on analysis of a structure of an existing product, and simulation of its design process.

Figure 17 shows the snapshots of the system while a designer carried out design of an indoor unit product family. a designer can edit and browse a product description on graphical user interfaces of the prescriptive models. In this example, a designer enumerates six customer needs of indoor unit product family; thermal management, humidity management, clean air, good looking and save energy. 'Thermal management' is developed into two customer needs; appropriate room size and quick warming/cooling (Figure 17-①). A designer picked up two customer needs as axes of market segment grid; good looking and appropriate room size. In the grid, six product

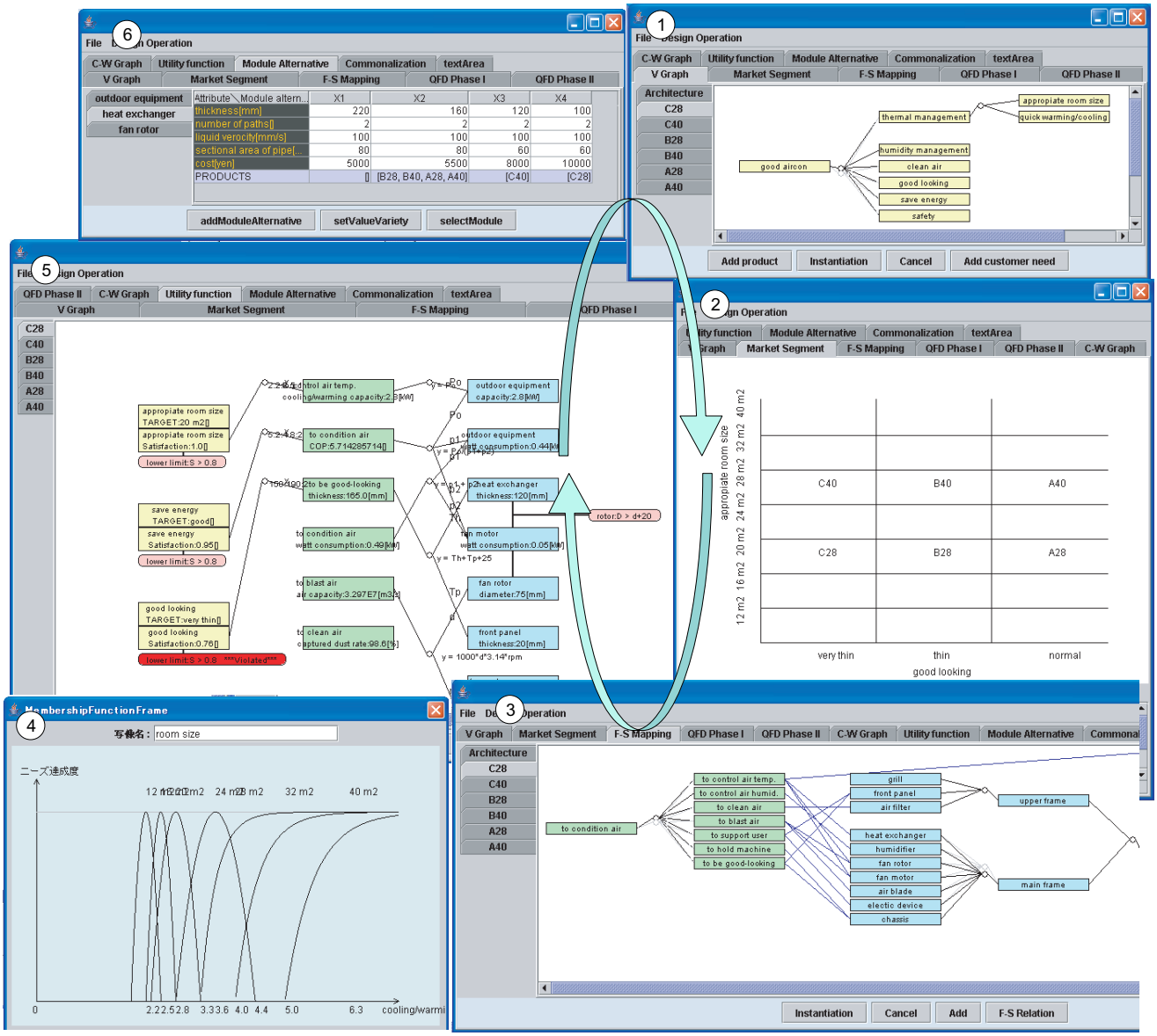

Figure 17. Prescriptive models used in design process

variants are planned; C28, C40, B28, B40, A28 and A40 (Figure 17-②). Functions and components of each product variant are designed by function-structure mapping (Figure 17-③). In order to evaluate a customer's satisfaction, utility function model is used (Figure 17-④). In this example, a satisfaction degree of 'good looking' of C28 does not meet the target 'very thin' (Figure 17-⑤). In order to thin an indoor unit, a thinner heat exchanger is required. A designer selected a thinnest heat exchanger module at module alternative browser (Figure 17-⑥), and reduced a diameter of a fan rotor which satisfies a size constraint in order to put a fan rotor inside of a heat exchanger.

This design process is automatically captured as a byproduct of a designer's operations on the system. Figure 18 shows a part of the captured design process in argumentation level. Issue nodes (blue node with question mark) and position nodes (red node with exclamation mark) are automatically generated. A gray node is a discarded position. An argument node (green node) is added by a designer to explain the branch of position nodes. Figure 18 shows that four positions is suggested for an issue of 'diameter of fan rotor of C28,' and that one position '75' is employed.

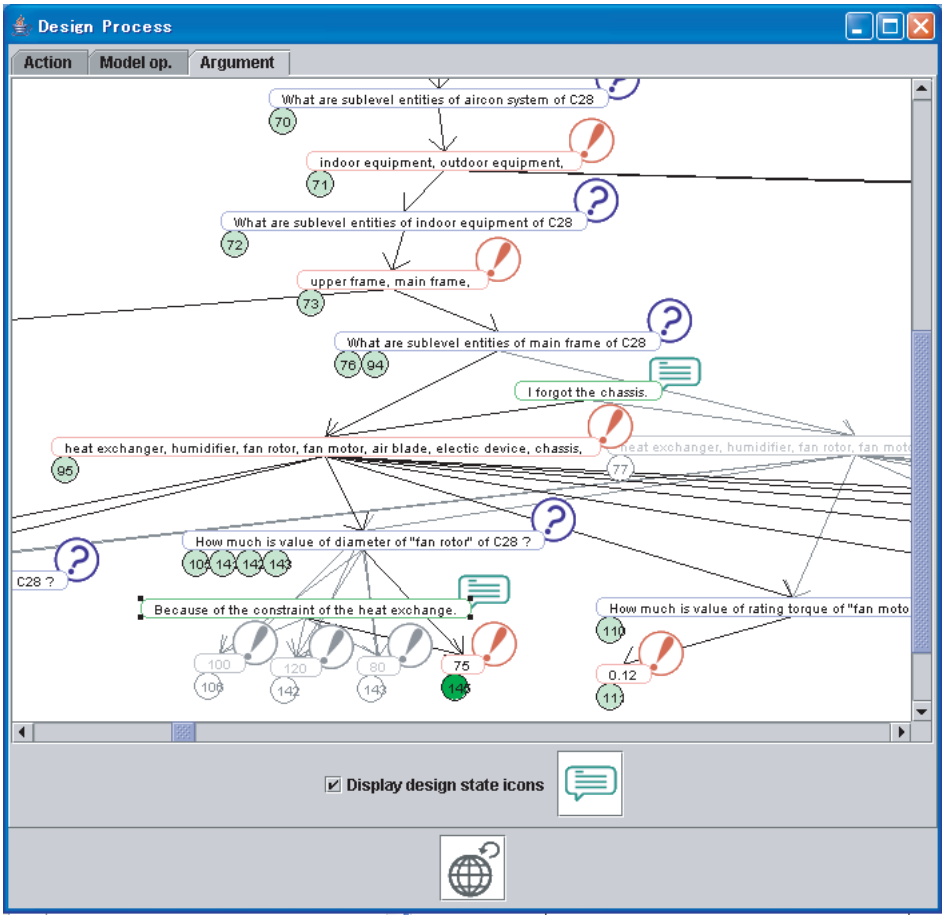

Figure 18. Captured argumentation

6 RELATED WORKS

A knowledge modeling method provides a necessary framework for a design description in all design domains. Many research groups have conducted this topic. In the domain of product architecture and family design, knowledge modeling of design is becoming a hot topic although its ability is still very limited (Simpson, 2004). Table 1 summarizes six works related to this research and compares their research aims and features of modeling ontology with ours.

The research group of NIST (National Institute of Standard and Technology, U.S.A.) has been developing a knowledge model called Core Product Model (CPM) toward large-scaled repository system of design rationale (Fenves, 2001). Wang *et al.* (2003) extended CPM to represent an evolution of product families and rationale of changes involved. They focus on design repository of rationale of product family deployment so that their modeling ontology includes concepts, such as version and series of product variants. However, they don't care about knowledge acquisition so much. The concepts about available module alternatives and commonalization of components are not considered, which are mainly used at design process rather than at the result of design.

A configuration framework for mass customization of products that employs UML is introduced by Felfernig (Felfernig *et al.*, 2001). They focus on knowledge acquisition and maintaining knowledge bases. A product description written in the UML format is automatically translated into an executable logic representation in order to employ model-based diagnosis techniques for debugging a faulty configuration knowledge base, detecting infeasible requirements and for reconfiguring old configurations. Their knowledge model represents structural architecture and module alternatives.

To reduce data redundancy when modeling families of products, the Generic Bill-of-Material (GBOM) concept developed at the Eindhoven University of Technology allows all variants of a product family to be specified only once (Erens and Hegge, 1994). McKay *et al.* (1996) combined the GBOM concept with product concepts and software to reduce

Table 1. Comparison of related works

		Nomaguchi *et al.* 2006	Wang *et al.* 2003	Felfernig *et al.* 2001	McKay *et al.* 1996	Claesson *et al.* 2001	Mortensen *et al.* 2005	Nanda *et al.* 2005
Aim	Design repository	X	XX	X	-	-	-	x
	Knowledge acquisition	X	-	X	X	x	x	x
	Design support by prescriptive model	X	-	-	x	x	x	x
Modeling ontology	Structural architecture	X	X	X	X	X	X	X
	Attribute architecture	X	x	-	X	-	-	-
	Module alternative	X	-	X	-	x	X	-
	Product variant	X	X	X	X	X	X	X
	Commonalization	X	-	-	-	-	-	-
	Attribute value	X	X	-	x	-	-	-
	Prescriptive model	X	-	-	-	-	-	x
	Others		Product evolution (version, series,..)					

XX; more considered than this research,
X; considered as well as this research,
x; considered but a little,
-; not considered.

data redundancy when considering multiple views, e.g., sales, manufacturing and assembly. Their knowledge model is useful for design repository and knowledge acquisition because it can represent structural architecture, attributes architecture and module alternatives.

Claeson *et al.* (2001) uses function-means-trees to create configurable components that represent a parameterized set of design solutions. This knowledge model was deployed at Saab automobile to help control product variety. Mortensen *et al.* (2005) proposed PFH (Product Family Hierarchy) diagram which visualizes structural architecture and a key component which could be a key of commonalization among product family. PFH is deployed at the Danish company Martin Professional A/S. Both knowledge models employ simple ontology of structural architecture and module alternatives. Although there is a limitation in evaluating designed architecture reflectively because the knowledge model represents neither prescriptive model nor attribute architecture, they succeeded in applying the knowledge model to company use.

Nanda *et al.* (2005) proposed a knowledge model based on OWL (Web Ontology Language) in order to capture, share, and organize product design contents concepts and contexts across different phases of the product design process. This knowledge model represents structural architecture of three aspects; customer needs functions and components. They integrate QFD to the knowledge mode.

The most advanced feature of the knowledge model proposed in this research is to support reflective process of designing product architecture and product family. This feature is enhanced by the framework of a knowledge management oriented design support system. However, in order to effectively utilize the framework, ontology should be carefully defined to meet the requirement of the framework; that is, describing any state of design and integrating prescriptive design models. Therefore, ontology defined in this research takes a crucial role to realize the design support system.

7 CONCLUSIONS

This paper proposes a knowledge model for a knowledge management oriented support system for product architecture and family design. The ontology of the knowledge model is defined in order to formally capture any design state of product architecture and family design, and to integrate prescriptive design models. The implemented system can support reflective process of designing product architecture and product family with the proposed knowledge model. The ability of the system is evaluated by performing the illustrative example design. Further studies should be performed to test the ability in practical cases which is more complicated and large-scaled.

ACKNOWLEDGMENTS

We appreciate Mr. Hiroyuki Itoh of DAIKIN Air-Conditioning And Environmental Laboratory, Ltd. for his cooperation to the air conditioner design example. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research, 17360072, 2005.

REFERENCES

Claesson, A., Johannesson, H. and Gedell, S., 2001, "Platform Product Development: Product Model a System Structure Composed of Configurable Components," *In Proceedings of the DETC'01 ASME 2001 Design Engineering Technical*

Conferences and Computer and Information in Engineering Conference, DTM-21714.

Doyle, J., 1979, "A Truth Maintenance System," *Artificial Intelligence*, Vol. 12, No. 3, pp. 231-272.

Erens, F. J. and Hegge, H. M. H., 1994, "Manufacturing and Sales Co-ordination for Product Variety," *International Journal of Production Economics*, Vol. 37, No. 1, pp. 83-99.

Felfernig, A., Friedrich, G. and Jannach, D., 2001, "Conceptual Modeling for Configuration of Mass-Customizable Products," *Artificial Intelligence in Engineering*, Vol. 15, pp. 165-176.

Fenves, S. A, 2001, "Core Product Model for Representing Design Information," *NISTIR 6736*, NIST, Gaithersburg, MD, 2001.

Fujita, K., 2002, "Product Variety Optimization under Modular Architecture," *Computer-Aided Design*, Vol. 34, No. 12, pp. 953-965.

Kunz, W. and Rittel, H., 1970, "Issues as elements of information systems," *Working Paper* No. 131, Berkeley, University of California, Berkeley, Institute of Urban and Regional Development.

Liao, S., 2003, "Knowledge Management Technologies and Applications – Literature Review from 1995 to 2002," *Expert Systems with Applications*, Vol. 25, No. 2, pp. 155-164.

Martin, M. V. and Ishii, K., 2002, "Design for Variety: Developing Standardized and Modularized Product Platform Architectures," *Research in Engineering Design*, Vol. 13, No. 4, pp. 213-235.

McKay, A., Erens, F. and Bloor, M. S., 1996, "Relating Product Definition and Product Variety," *Research in Engineering Design*, Vol. 8, No. 2, pp. 63-80.

Meyer, M. H., 1997, "Revitalize Your Product Lines Through Continuous Platform Renewal," *Research Technology Management*, Vol. 40, No. 2, pp. 17-28.

Mortensen, N. H., Munk, L. and Fiil-Nielsen, O., 2005, "Preparing for a Product Platform - Product Family Hierarchy Procedure -," *In Proceedings of International Conference on Engineering Design (ICED 05)*, 296.45.

Nanda, J., Simpson, T. W., Shooter, S. B. and Stone, R. B, 2005, "A Unified Information Model for Product Family Design Management," *In Proceedings of the DETC'05 ASME 2005 Design Engineering Technical Conferences and Computer and Information in Engineering Conference*, 84869.

Nomaguchi, Y., Ohnuma, A. and Fujita, K., 2004, "Design Rationale Acquisition in Conceptual Design by Hierarchical Integration of Action, Model and Argumentation," *In Proceedings of the 2004 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, CIE-57681.

Nomaguchi, Y. and Fujita, K., 2005, "Product Architecture Design Process for Model-based Knowledge Management," *In Proceedings of 15th International Conference on Engineering Design (ICED 05)*, 146.81.

Rosenman, M. and Simoff. S. J., 2001, "Conceptual Modeling in Design (Special Issue)," *Artifical Intelligence in Engineering*, Vol. 15, No. 2.

Schön, D. A., 1982, *The Reflective Practitioner – How Professionals Think in Action*, Basic Books Inc.

Simon, H. A., 1969, *The Sciences of the Artificial*, The MIT Press.

Simpson, T. W., 2004, "Product Platform Design and Customization: Status and Promise," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 18, No. 1, pp. 3-20.

Simpson, T. W., Siddique, Zahed and Jiao Jianxin (editors), 2006, *Product Platform and Product Family Design*, Springer Science+Business Media, Inc.

Ulrich, K., 1995, "The role of product architecture in the manufacturing firm," *Research Policy*, Vol. 24, No. 3, pp. 419-440.

Wang, F., Fenves, S. J., Sudarsan, R. and Sriram, Ram. D., 2003, "Towards Modeling the Evolution of Product Families," *In Proceedings of the DETC'03 ASME 2003 Design Engineering Technical Conferences and Computer and Information in Engineering Conference*, CIE-48216.

Wilhelm, B., 1997, "Platform and Modular Concepts at Volkswagen — Their Effect on the Assembly Process," *Transforming Automobile Assembly: Experience in Automation and Work Organization*, K. Shimokawa, U. Jürgens and T. Fujimoto, eds., Springer-Verlag, New York, pp. 146-156.