Turk J Elec Engin, VOL.9, NO.1 2001, © TÜBİTAK

An Integrated Design-Object Modeling Environment – Pluggable Metamodel Mechanism –*

Masaharu YOSHIOKA
National Institute of Informatics
Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo, 101-8430-JAPAN
Takayuki SEKIYA, Tetsuo TOMIYAMA
Research into Artifacts, Center for Engineering,
The University of Tokyo
Komaba 4-6-1, Meguro-ku, Tokyo, 153-8904-JAPAN

Abstract

We propose a new integrated computational environment to support design object modeling, during engineering design process that requires a variety of design object models, such as a geometric model, a control model, and a finite element model. To integrate multiple design object models, we have developed a mechanism called a metamodel mechanism that maintains consistency among various models. The metamodel mechanism represents relationships among concepts used in these models and is useful, for example, to conduct concurrent engineering practices. In this framework, we formalize a design process as operations to the metamodel. This paper expands the concept of the metamodel mechanism to allow plugging in external design object modelers. We call it a pluggable metamodel mechanism on which design process knowledge navigates a design process by choosing the most appropriate design object modeler. Finally, we describe a prototype system and illustrate an example design.

Key Words: *Design Process, Model Integration, Model-based Reasoning*

1. Introduction

Concurrent engineering aims at taking a variety of product life cycle issues, such as manufacturing, operations, maintenance, and recycling into the design stage of products. This requests to develop computational tools to model and evaluate these life cycle stage issues during the design stage. Intelligent CAD was initially proposed to overcome problems associated with conventional geometry-based CAD [1], and has a potential to deal with a variety of product life cycle knowledge as a knowledge intensive design environment. However, our experiences with building such a large scale knowledge base [2] tells that systematization of design knowledge is indispensable even only to collect knowledge [3]. This means that design knowledge must be systematically formalized, made computable, and organized in the knowledge base to achieve flexible, efficient, effective reuse and sharing of knowledge. Here, by design knowledge we mean collective product life cycle knowledge.

*An earlier version of this paper was published in "Computer Aided Conceptual Design '97," Alan Bradshow and John Counsell (Eds.), Lancaster University, 1997.

In an engineering design process, designers use various kinds of design object models. Since these models are related to each other (e.g., shape data of a finite element model is closely related to shape data of a solid model), we need an integrated design object modeling environment to manage these models.

We have already proposed a Knowledge Intensive Engineering Framework (KIEF) that is a computational framework to integrate these design object models (Figure 1) [4]. KIEF employs a *metamodel mechanism* [5, 6] which represents and maintains relationships (e.g., causal dependency, translation, attribute, etc.) among concepts used in these models (Figure 2).

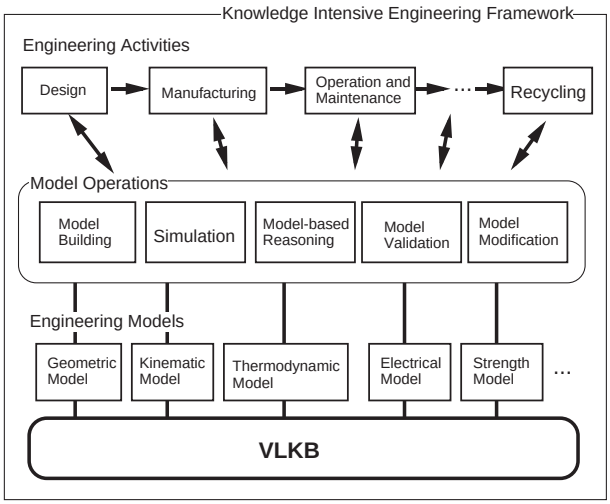

Figure 1. The Knowledge Intensive Engineering Framework (KIEF) [4]

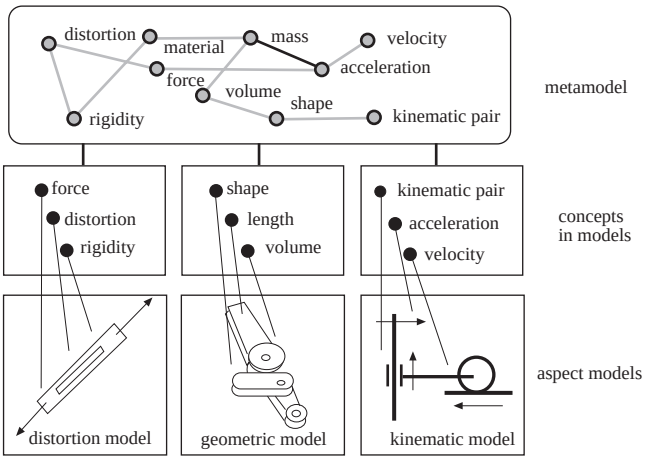

Figure 2. The Metamodel Mechanism [4]

Nowadays, many commercial design object modelers are available to build, modify, operate and evaluate these object models. In this paper, we further propose a *Pluggable Metamodel Mechanism* that allows to plug in existing external design object modelers. To do so, we need a schema to describe what a modeler can do and what kind of data is required for it to build a model. The pluggable metamodel

mechanism should allow data exchange among design object modelers as well.

In this paper, we regard a modeling process as operations to the metamodel. There are two steps in building a design object model. In the first step, a designer makes a qualitative model which represents relationships among concepts used in the model. In this step, the designer sometimes abstracts the object and simplifies the model. After building this qualitative model, in the second step, the designer checks other models that represent the same object to obtain quantitative data about the object by exchanging data among these models.

When many modelers are available to designers, it becomes difficult for them to select an appropriate model in evaluating the design object. Therefore, KIEF should support the designers to select appropriate models by using design process knowledge.

As a result of our previous research on design process knowledge, we have proposed a computable design process model based on analysis of design protocols [7]. This model has two levels of inferences: an action level and an object level. In this model, a design object is described in first order predicate logic in the object level, while a design process is represented and navigated by a sequence of design operations including modification and evaluation of the design object model in the action level. In other words, the design process knowledge is operational knowledge about the design object knowledge. In this paper, we expand this design process model to handle various design object models, which are plugged to the pluggable metamodel mechanism.

A number of research efforts have been made to integrate various design object models. IGES (Initial Graphics Exchange Specification) [8] aims to integrate graphical data among different CAD systems. However, IGES deals with only graphical data. Therefore, research in STEP (Standard for the Exchange of Product model data) [9] started. STEP aims at setting up a product model standard for exchanging data from one CAD system to another. STEP is intended to handle all kinds of data which are used during the whole life cycle of a product. Nevertheless, STEP in its current stage only is used to support data exchange.

This paper is divided into six sections. Section 2 illustrates a framework of the metamodel and our techniques to deal with functional information. We also introduce the computable design process model. In Section 3, we propose the pluggable metamodel mechanism. We also discuss a formalization of a modeling process based on this mechanism, a schema to describe plugged-in tool, and a methodology to transfer data among plug-in design object modelers. Section 4 discusses an extension of the computable design process model to deal with multiple design object models. Section 5 illustrates a prototype system of the pluggable metamodel mechanism based on the computable design process model. Section 6 compares our approach with other related work and Section 7 concludes the paper.

2. Models of Design Processes and Design Objects

2.1. The Metamodel Mechanism

A design object can usually be represented with various aspect models, that encompass different domains to be considered. For instance, domains include function, geometry, and kinematics. A functional model represents the designer's intention, a heat model represents heat flow of the design object, and a kinematic model represents motion of the design object. These aspect models are not independent of each other; therefore, we need to have a mechanism to maintain consistency among aspect models.

The *metamodel mechanism* is a modeling framework for integrating multiple aspect models [5, 6]. It has symbolic representation of concepts about physical phenomena and mechanical components (Figure 2). A *metamodel* of a design object is represented as a network of relationships among concepts that appear in the aspect models. Types of relationship include, for example, causal dependency among physical phenomena, arrangements of components, and attributive relationships.

The designer builds an initial qualitative model of the principal physical behavior and the structure of the design object. We call this initial model a *primary model*. A primary model is an aspect model that represents the designer's mental model about behavior. An initial metamodel is generated from the primary model through qualitative reasoning based on QPT (Qualitative Process Theory) [10] by envisioning possible physical phenomena. This is computed by a qualitative reasoner [6]. After building this metamodel, the designer can generate qualitative aspect models by selecting concepts which are related to the aspect.

2.2. Functional Representation

Modeling function of a design object is of particular importance to assist the designer in the conceptual design. A functional model is an aspect model that can represent the designer's intention.

We use the FBS (Function-Behavior-State) diagram [11] to represent functions. An FBS diagram denotes two types of relationships. One is relationship between behavior and state. A state is described by entities, attributes, and relations among them. Behavior is a sequence of one or more changes of states. This relationship between behavior and state is called *B-S relationship*. The other is relationship between function and behavior. A function is a subjective description of behavior abstracted through human recognition of the behavior in order to utilize it. This relationship between function and behavior is called *F-B relationship*, and gives physical semantics to a function.

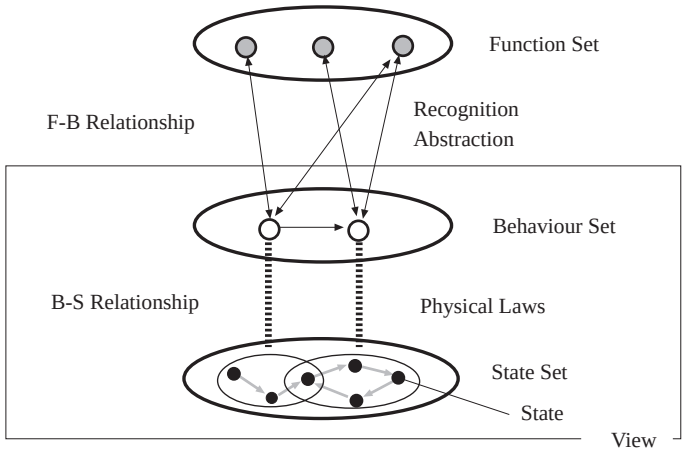

Figure 3. FBS Diagram [11]

An FBS diagram of a design object is composed of functional prototypes that represent the F-B relationships and physical features that represent behaviors and states of the design object. Table 1 shows the scheme of a functional prototype [11].

The designer uses the FBS diagram in two ways during functional design. One is function decomposition in which a function is divided into subfunctions. The other is function synthesis in which physical features that can exhibit a desired function are searched for and instantiated.

Table 1. Definition of a Functional Prototype

Item	Contents
Name	*verb + objectives*
Decomposition	networks of subfunctions
F-B Relationship	physical features

The metamodel mechanism maintains the consistency of the B-S relationships. Feasibility of functions is tested against the specification by reasoning out behavior of the design object with QPT based modeling system.

2.3. The Computable Design Process Model

Takeda *et al.* [12] observed from design experiments that a design object evolves in a step-wise manner. One step consists of five elementary stages; i.e., *awareness-of-problem, suggestion, development, evaluation,* and *conclusion*. These five stages form a cycle called a *design cycle*. A design cycle solves a small design problem or divides it into smaller subproblems. This observation from design experiments led to a cognitive model of design processes which is repetition of design cycles (Figure 4).

We have formalized the design cycle with logic [7] (Figure 5). Suggestion in a design cycle is a stage in which the designer suggests new candidate design solutions for a design problem, and it can be logically simulated by a process of *abduction*. Abduction, proposed by Peirce [13], is a type of inference to find premises from given axioms and theorems. Development and evaluation are stages in which the designer figures out the properties of a candidate design solution and evaluates the properties with respect to the specifications. Development is much similar to simulation to see how a candidate behaves under a certain circumstance. We can formalize development and evaluation as processes of *deduction*.

The designer often faces inconsistency of information among different models during the design process. In most cases, inconsistency means that knowledge is incomplete in that sufficient descriptions about the applicability of the knowledge are not provided. Circumscription [14] can find implicit descriptions of the knowledge to restrict its applicability and modify the knowledge to resolve the inconsistency.

The action level inference guides the entire design process and is considered meta level inference. In the action level, actions to be taken in the next step are derived from the current situation of the object level. Figure 5 depicts this computable design process model. Knowledge used in the object level is chosen by the action level inference based on the action level knowledge and the current situation of the object level influence. Once a design cycle successfully completes, knowledge for the next design cycle is made available by the action level inference.

3. The Pluggable Metamodel Mechanism

The metamodel mechanism described in Section 2 requires describing methods to exchange data among aspect models through one to one correspondence; e.g., the length of a beam is directly computed from solid data. However, it is not an easy task to describe all correspondences about the modelers plugged in. To avoid this problem, we here propose a new framework, called a *Pluggable Metamodel Mechanism*, that allows to plug in existing external design object modelers without such one-to-one correspondences.

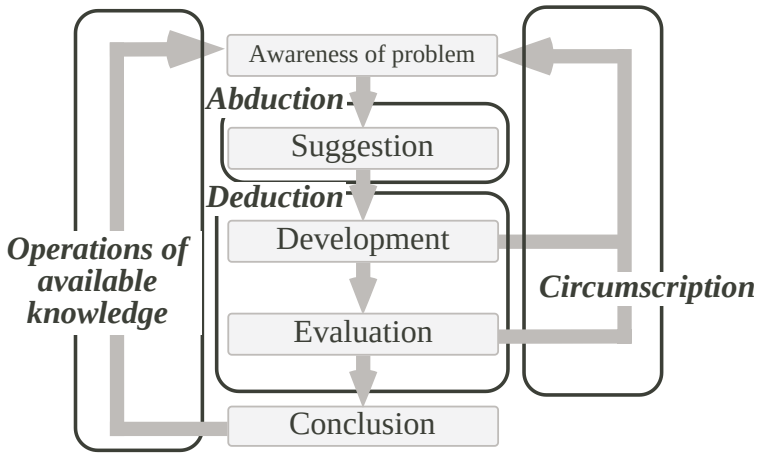

Figure 4. Logical Inferences and the Cognitive Design Process Model [7]

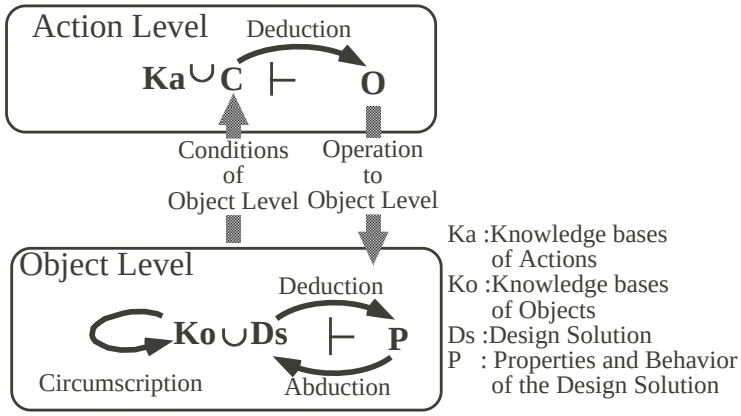

Figure 5. The Computable Design Process Model [7]

3.1. Knowledge Representation

The pluggable metamodel mechanism as a kernel of KIEF is requested to have a capability of handling a wide variety and a huge amount of design knowledge. To handle such a large amount of knowledge, traditionally engineering was organized as micro theories capable of generating and verifying a new solution. These micro theories tend to be dealt with just as a black box. For example, in the design of a robot, a micro theory about infrared sensors can be introduced. In this case, however, designers do not need to know details about behavioral principles of such sensors. All they should know for using a sensor is the relationship between its input and output.

The pluggable metamodel mechanism can deal with multiple micro theories by associating with a model and knowledge about how to use them. In order to achieve integration of the results from each modeling system, the mechanism should be equipped with common ontology and general knowledge about physical concepts.

Figure 6 depicts the component architecture of the knowledge base system with an example of gear transmission. The middle component, called *concept base*, contains a basic ontology about physical concepts. In the concept base, physical concepts are organized as super (abstract) - sub (concrete) hierarchies and categorized into the following four types.

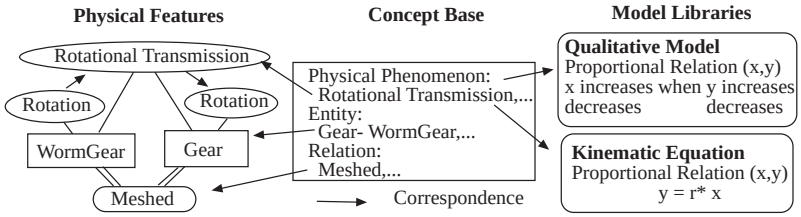

Figure 6. Knowledge Architecture for the Pluggable Metamodel Mechanism

- Entity
  An entity represents an atomic physical object. Entities include, e.g., mechanical parts and electric devices, and are organized in an abstract- concrete hierarchy. For example, a "worm gear" is a subclass of a "gear." The hierarchy allows multiple inheritance. In addition, entities have descriptions of "has" relationships with related physical properties.

- Relation
  Relations represent relationships among entities to denote static structure. They include relations between physical objects such as "connection" and "on." They are also organized in an abstract-concrete hierarchy.

- Attribute
  An attribute, such as "position" and "temperature." is a concept attached to an entity and takes a value to indicate the state of the entity. Attributes also have a description about differential relationships with other attributes (e.g., "velocity" is differential of "position").

- Physical phenomenon
  A physical phenomenon designates physical laws or rules that govern behaviors. A physical phenomenon is associated with related physical phenomena, entities, and attributes with respect to the phenomenon. It is also associated with physical rules for representing relationships among attributes.

The left side of Figure 6 contains a "physical feature," such as a worm gear pair, which represents a combination of a set of entities and relations among the entities, and physical phenomena causally related to the entities. Physical features are used as building blocks for a physical model on this system. Physical concepts in the concept base provide a vocabulary to build the physical feature in this basic ontology layer. Model fragments for building a model with various model representations are stored in the model libraries with the relationship to physical phenomenon.

3.2. Exchanging data among design object models

In order to support data exchange processes with the pluggable metamodel mechanism, we should take the following two points into consideration.

- Representation of knowledge and information in the metamodel.
  There can be different kinds of representation methods (data structure) for each attribute (e.g., solid, face, force, etc.). The system should absorb the differences in representation.

- Selection of an appropriate modeler.
  Since we do not directly describe the correspondence among modelers to exchange data, the system should select an appropriate modeler and method to obtain data from the modeler.

3.2.1. Representation of data

To absorb different data structure, we should define standard data structure for each attribute. However, some attributes, which have complex data structure, are difficult to set these standard definitions (e.g., solid, free surface, etc.). Therefore, we define the standard in two ways.

- Standard methods to obtain data
  For some attributes that have complex data structure, instead of dealing the data structure itself, we define standard methods to obtain simple structured data (e.g., getting attribute information of a vertex from the information of a solid). This task can be made easier if we use a product data model such as STEP.

- Standard data structure
  For other attributes that have simple data structure, such as vertex, we define standard data structure.

By using these standard methods, the pluggable metamodel mechanism can obtain and exchange data between plugged-in modelers.

3.2.2. Selection of an Appropriate Modeler

During a modeling process, some data might be missing. This data can be obtained from other aspect modelers. To select an appropriate aspect model, we assume that the following three descriptions about aspect modelers are needed.

- Concepts used to build an aspect modeler
  This description is necessary to generate an qualitative aspect model from the metamodel.

- Data handled by an aspect modeler
  This description is used to identify an appropriate aspect model, when there is missing data.

- Data exchange method
  This description is used to feed data from a source attribute model to the aspect modeler.

Table 2 defines the knowledge about aspect modelers. We use available ontology to generate a qualitative aspect model by abstracting the metamodel with these concepts. Computable ontology is used to find out an appropriate aspect modeler for computing the value of attributes. We also define attribute translation methods from attribute(s) in the metamodel to an attribute in the modelers. These methods are represented by a conceptual network used to find out source attribute(s) from the metamodel by pattern matching of the graph. In addition, we define related ontology for filtering out unrelated concepts from the metamodel during construction of the initial qualitative aspect model. Table 3 is an example description about a beam modeler.

These concepts are all defined in the concept base that we described in Section 3.1. Because of this, when force is focused on, we can deal with any force regardless of its origin.

Table 2. Knowledge about Modelers

Name of the slot	Contents
Related ontology	List of concepts
Available ontology	List of concepts
Computable ontology	List of concepts
Data exchange method	Attribute relationship graph and translating method

Table 3. Knowledge about Beam Modeler (Example)

Name of the slot	Contents
Related ontology	Entity, Relation, Force
Available ontology	Beam, HingedSupport, ConcentratedForce, etc.
Computable ontology	ShearingForceDiagram, BendingMomentDiagram
Data exchange method	See Figure 7 for example

Figure 7 shows the definition of an attribute translation method. The target attribute list (the upper part of center) represents the attribute list that is used in this beam modeler. The method list (beneath the target attribute list) represents candidate translation methods to obtain the selected target attribute (e.g., in this case "ForceTransmissionPoint" of "TransmittedForce"). The required concept graph (right upper corner) represents the required attribute value to exchange data. In this case to calculate the attribute value "Force TransmissionPoint," we need the attribute value of "Line2D" which is an attribute of relation "Relation" between "Beam" and "Entity" where the "Transmitted Force" comes from. In the method browser, we define a data translation method as a Smalltalk program.

3.3. Formalization of the Modeling Process

Since designers obtain and modify data of a design object that is already defined in the modeling process, it is necessary to formalize this modeling process for supporting data exchange among aspect models.

We formalize this modeling process as follows.

1. *Determine abstraction level for a design object modeler* (Figure 8-(a)).
   The metamodel mechanism determines the most appropriate abstraction level for the selected modeler based on the knowledge about the design object modeler. The metamodel mechanism suggests concepts included in the determined abstraction level and needed for the selected modeler, and the designer has to give abstract descriptions of the design object using these concepts. For example, consider the design of a robot arm. If the metamodel mechanism reasons out that distortion is possibly to happen, the metamodel mechanism suggests to describe the arm only with concepts such as beam, load, support, and bending. Then the designer abstracts the arm concept in the domain of geometry as a beam. This abstraction is computationally done by unification that is an operation to create a new instance that delegates the two concepts of shape "arm" and physical feature "beam."

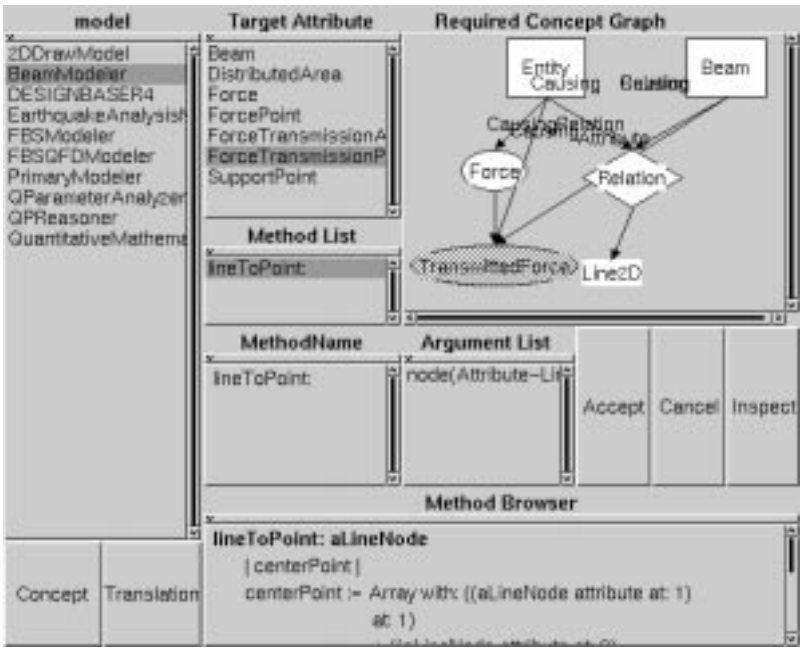

Figure 7. The Modeler Knowledge Browser

2. *Determine simplification level for the selected modeler* (Figure 8-(b)).
   The designer determines an appropriate simplification level for the selected modeler. The designer selects physical phenomena that should be considered in the selected aspect model. In other words, some physical phenomena and some entities might be neglected. For instance, the designer can say that any bending is considered, while vibration is not of interest in the arm design.

3. *Exchange data among aspect models* (Figure 8-(c)).
   The metamodel mechanism generates aspect models of the design object from the metamodel. Suppose in the example of the arm design, a bending aspect model is generated. Since aspect models often require numerical information, the metamodel mechanism requests the designer to specify an appropriate aspect model to feed the required numerical information. For instance, since the generated bending aspect model needs dimensions of the beam, the designer specifies a solid modeler as a source to provide geometric information about the arm. Once this information is given, the metamodel mechanism can consistently maintain the relationships between these two models.

4. Model Operations in the Computable Design Process Model

When many modelers are plugged into KIEF, it becomes difficult for the designer to select appropriate models to evaluate the design object. Therefore, we assume KIEF should support designers to select appropriate models by using design process knowledge.

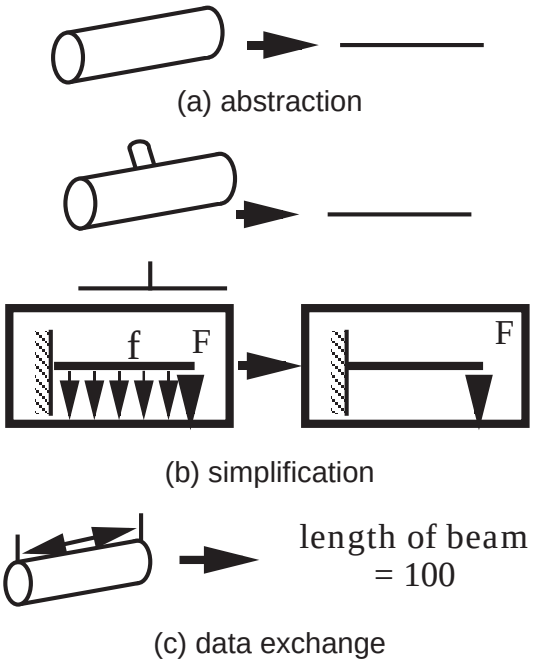

Figure 8. Modeling Process

4.1. Extending the Computable Design Process Model

In the design process model described in Section 2.3, we assumed that there was only one aspect model through a whole design process. We extend the computable design process model by incorporating the pluggable metamodel mechanism as object level representation. Since we assume functional information and qualitative information are essential in early stages of the design process, an FBS modeler, which can handle FBS diagrams, and the qualitative reasoning system are plugged into the pluggable metamodel mechanism.

Abduction in the extended computable design process model is realized as operations to the FBS modeler to find out a candidate that exhibits the most appropriate behavior from the given functional specifications. During functional design, the designer builds the functional structure of the design object with physical features from the functional specification. If the specification is too abstract to be associated with physical features, the designer recursively decomposes the specification into subfunctions. The functional structure is completed, when all the decomposed subfunctions are embodied by physical features that realize as a whole the specification.

Deduction in the extended design process model performs simulation of behavior and is realized as model operations to the pluggable metamodel mechanism. This process involves two steps. One is a process corresponding to the development of the design cycle in which a metamodel is recomputed by the qualitative reasoning system from the primary model built during the functional design. The other corresponds to the evaluation of the design cycle to evaluate the design object using multiple aspect models. Therefore, model operations include those to build a primary model and those that deal with multiple aspect models.

4.2. Dealing with Aspect Models in the Computable Design Process Model

In the extended design process model (Figure 9), the designer modifies and evaluates the design object using multiple aspect models. Selecting an appropriate aspect model is equivalent to selection of available

knowledge in the design process model. Switching one aspect modeler to another implies changes of the designer's focus in the design object.

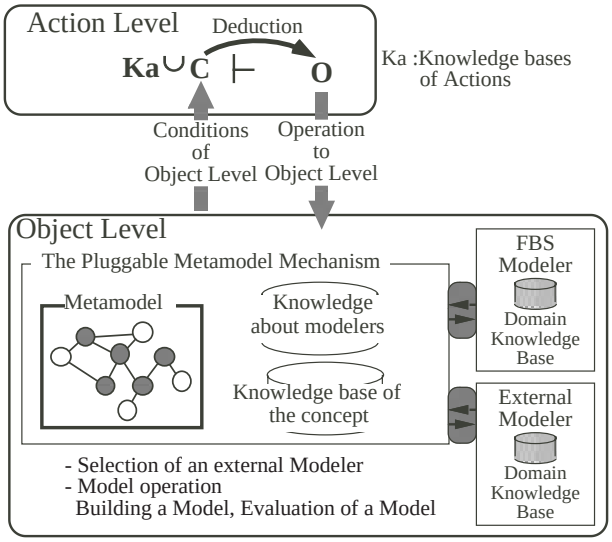

Figure 9. The Extended Design Process Model

We formalized this selection of an aspect model as follows.

1. *Select an appropriate design object modeler.*
   Selection of a design object modeler depends on the design context triggered by, for example, execution of a modification operation. For instance, the designer adds new descriptions of the design object using a functional model, and updates the metamodel according to the progress of the design process. Therefore, the action level inference selects the qualitative reasoning system to recompute all physical phenomena that possibly occur. These physical phenomena and the changes of the metamodel define a new design contexts according to which the action level inference selects an appropriate aspect model.

2. *Generate and evaluate aspect models.*
   After selecting an appropriate design object modeler, the designer builds a new aspect model on the modeler and evaluates it. Newly obtained information about the design object can be propagated to other aspect models for further evaluation.

4.3. Consistency among Aspect Models

Since the designer evaluates aspect models with various independent domain knowledge, he/she is sometimes confronted with inconsistency. For example, a motor seems to rotate qualitatively, though it does not quantitatively because of friction.

In the extended design process model, the metamodel mechanism maintains the consistency among aspect models. Aspect models have information about assumptions which are used in building themselves. When a new aspect model is generated by the metamodel mechanism, the mechanism maintains the consistency between the new aspect model and other aspect models by checking assumptions of the new aspect model to see if they contradict with assumptions of other aspect models or not.

In case of the motor example, the pluggable metamodel mechanism can generate two types of aspect models; a model in which motor rotates, and otherwise.

5. The Prototype System

5.1. The System Architecture

Based on the concept of the pluggable metamodel mechanism and the extended design process model discussed in Sections 3 and 4, we have developed a prototype system. The system is implemented in VisualWorks[1]\Smalltalk on a Sun[2] workstation.

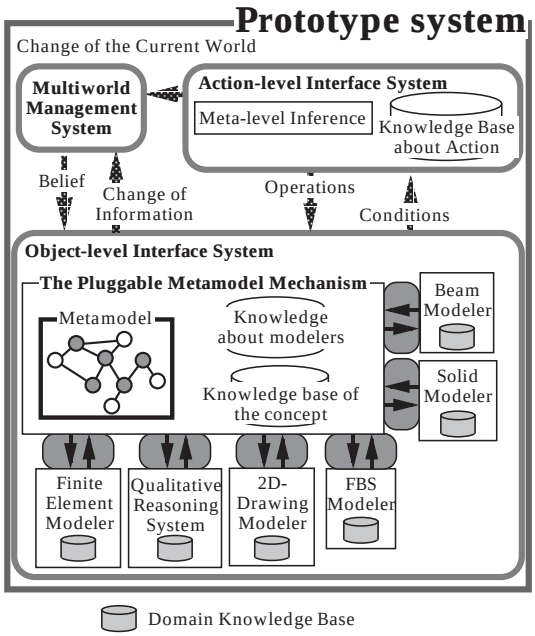

Figure 10. The Architecture of the Prototype System

Information in the object level is stored in the pluggable metamodel mechanism. Figure 10 depicts the architecture of the system. The current implementation of the prototype system is connected to six external design object modelers; i.e., a qualitative physics reasoning system [6], ProEngineer[3] (a solid modeler and its FEM preprocessor), a 2D draw modeler, an FBS Modeler, and a beam modeler whose inference engine is Mathematica[4] (a symbolic processing mathematics system).

The action level inference guides the designer to use the metamodel mechanism and plugged modelers (e.g., the FBS modeler, etc.) for abductive and deductive reasoning. The knowledge about the action level which does not depend on design objects is written in first order predicate logic. For example, the knowledge about development and evaluation is listed below in an abstract but extremely general way.

[1] VisualWorks is a registered trademark of Cincom Systems.
[2] Sun is a registered trademark of Sun Microsystems, Inc.
[3] ProEngineer is a trademark of Parametric Technology Corporation.
[4] Mathematica is a trademark of Wolfram Research, Inc.

if `(abducted)`
; if abduction takes place
then `(getNewAssumption NA)`
; then identify modified place in the design solution
`(makeMetamodel NA NewDerived)`
; and recompute a metamodel based on modification
`(selectAppropriateModeler NewDerived Modeler)`
; and select an appropriate modeler
`(generateAspectModel Modeler)`
; and generate an aspect model.

In general, there are three types of knowledge in the action level knowledge base; one for requesting specifications, one for suggesting candidate solutions, one for developing and evaluating solutions. The action level inference navigates model operations using these three types of knowledge.

5.2. Example

Let us illustrate an example of designing a warehouse cell of "Cellular Manufacturing System" that our group has developed [15] (Figure 11-(a)). This system consists of manufacturing cells and homogeneous intelligent autonomous warehouse cells (Figure 11-(b)).

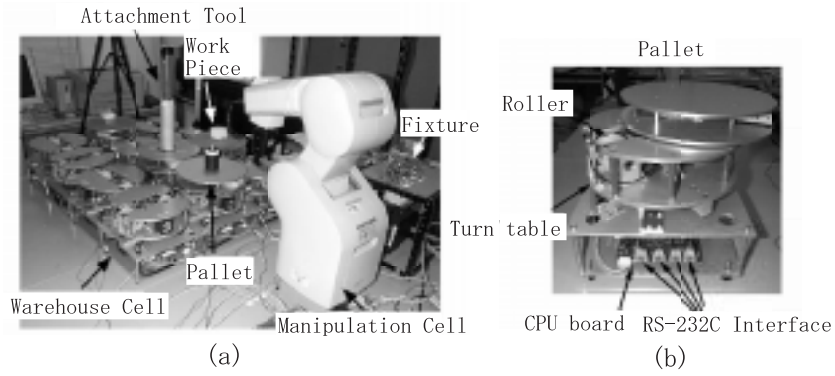

Figure 11. The Cellular Automatic Warehouse

We demonstrate how the system performs one design cycle in the design of a warehouse cell (Figure 12). Before starting the design cycle, we assume that the designer has already built the functional hierarchy of the cell and decided to use "Turning table" for achieving "Rotate(solid)" function. Figure 13 depicts the FBS diagram of the cell at the beginning of the design cycle. The oval nodes represent functions and the links between the oval nodes represent the functional hierarchy. The rectangular nodes denote physical features that compose the primary model for the metamodel mechanism.

After the designer decided to focus on the function of "move horizontally,' the system suggests two candidates to achieve this function; one is a roller and pallet mechanism and the other is a belt conveyer. The designer chooses the roller and pallet mechanism. Then the system adds a roller, a pallet, support structure and relations among them to the FBS diagram (Figure 14).

For selecting appropriate aspect models for evaluation, the pluggable metamodel mechanism uses the qualitative physics reasoning system [6] by reasoning about all possible physical phenomena that the roller

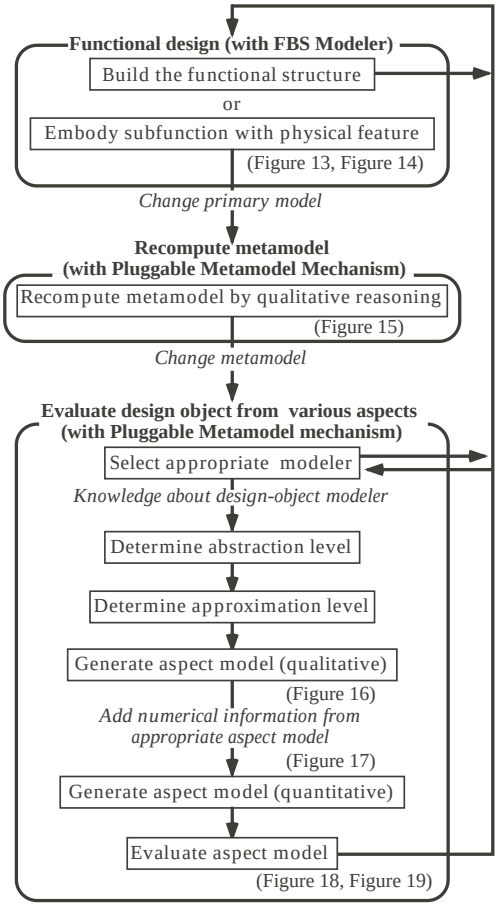

Figure 12. Design Cycle of the Prototype System

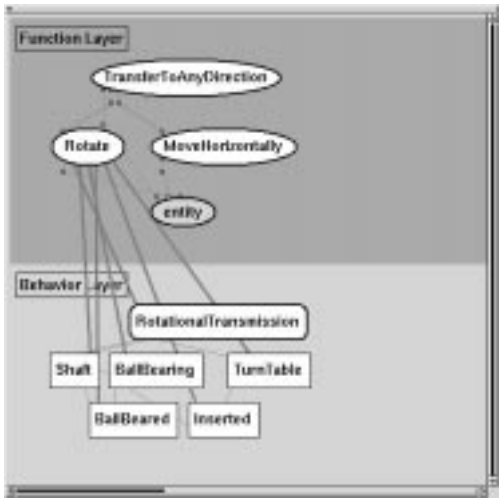

Figure 13. FBS Diagram of the Cell

and the pallet may cause. In Figure 15, icons with dotted areas represent the derived physical phenomena. Since the bending deformation occurred on the "Turn Table" that is also a "Case" of the "Roller," the system selects the aspect of strength of material to analyze deformation of the "Turn Table." Then a beam model and a finite element model can be generated. Note that, at this stage, these models are only qualitative and have no quantitative information.

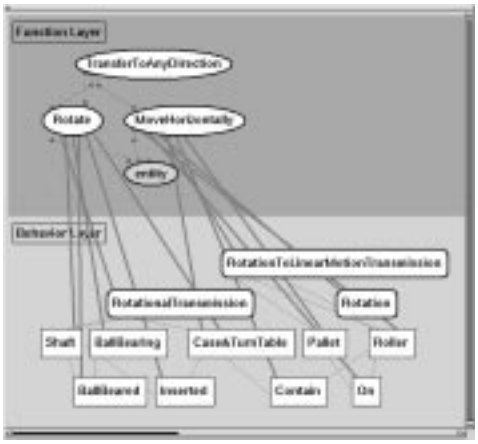

Figure 14. Revised FBS Diagram of the Cell

The designer selects the beam model to analyze deformation of the "Shaft." Then the pluggable metamodel mechanism prepares assumptions regarding abstraction of components to build the beam model; i.e., beam, load, support, and so on (Figure 16). However, at this stage there is no numerical information about the structure, the pluggable metamodel mechanism cannot generate the beam model. Therefore, the system selects the appropriate modelers to handle these numerical information (length of beam, position of support, etc.). In this case, the system selects a 2D draw modeler and a solid modeler by using the knowledge about modelers. Afterwards, designer selects a solid modeler to input shape data (Figure 17). After inputing shape data, the system starts to exchange data to the beam modeler. Since it is difficult to set the axis of beam automatically in general, the system requests the designer to set the axis of beam. Finally, a new beam model is generated. Figure 18 depicts the shearing force diagram of the shaft computed by Mathematica. Figure 19 shows the meshed data for FEM through a similar procedure.

6. Related Work

CDME (Collaborative Device Modeling Environment) [16] is another framework for integrating existing external modeling systems. It is similar to our approach in that integrating models takes place through translating their knowledge to one format; i.e., they use KIF (Knowledge Interchange Format) [17], Ontolingua [18] and CML [19], while we use QPT [10] as the most fundamental description. However they do not mention about the design process management, although an intelligent agent oriented architecture is assumed.

Xue *et al.* [20] developed an earlier version of IIICAD (Intelligent Integrated Interactive CAD) to integrate knowledge about design processes and objects. The integration of IIICAD is, however, less mutual and less closer than the system proposed here, in that the design process manager cannot deal with operations to aspect models.

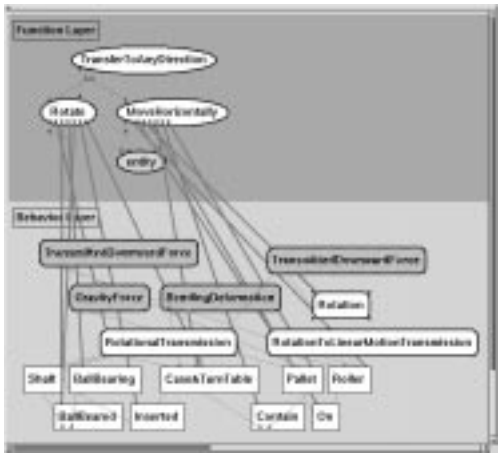

Figure 15. Result of the Qualitative Physics Reasoning System

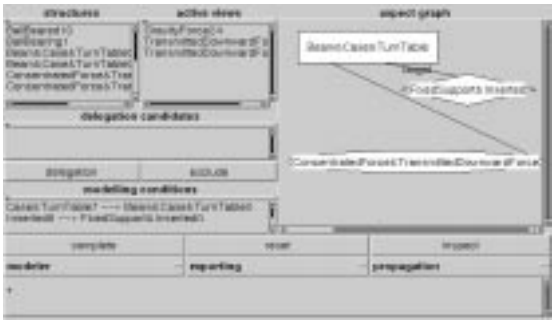

Figure 16. Generated Beam Model of the Cell

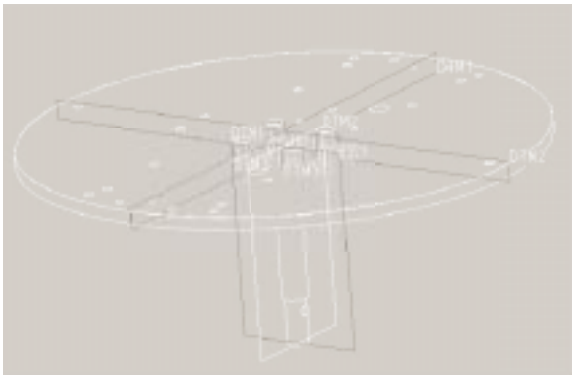

Figure 17. Geometric Model of the Turn Table and shaft

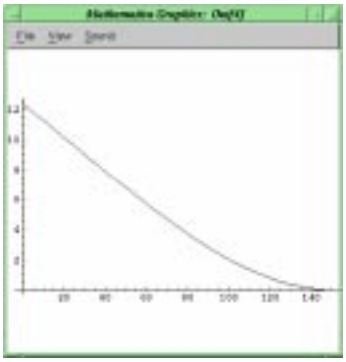

Figure 18. Result of Shearing Force Analysis of the Beam Model

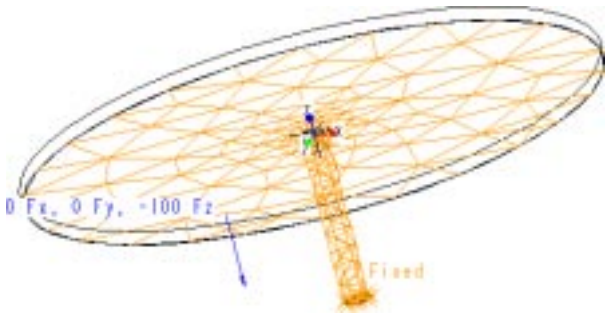

Figure 19. Generated Mesh Data for FEM

Nayak [21] proposed a method to select an adequate model which can explain the given problem. His approach is suitable to generate explanations that arrive at an expected conclusion. However, this is not the case for design, in which no conclusion can be assumed before analyzing behavior of a design object.

7. Conclusions

This paper described a new integrated environment for design object modeling. This environment has an ability to plug-in existing design object modelers and support modeling process with (1) symbolical operation to the model (abstraction, simplification), and by (2) exchanging data among models. In addition, design process knowledge navigates the designer to select appropriate models for evaluation. These appropriate aspect models are selected by focusing on physical phenomena that might occur to the design object. We think this framework can contribute to advances of product modeling environment and intelligent CAD architecture.

Future work includes plugging-in various tools to check the capability of this framework, and developing a method to maintain the consistency of quantitative data.

Acknowledgments

This research was partially supported by JSPS-RFTF 9600701, entitled “Modeling of Synthesis”, currently conducted at the University of Tokyo, the University of Osaka, Nara Institute of Science and Technology, and National Institute of Informatics, Japan.

References

[1] T. Tomiyama. Intelligent CAD systems. In G. Garcia and I. Herman, editors, *Advances in Computer Graphics VI, Images: Synthesis, Analysis, and Interaction*, pp. 343–388. Springer-Verlag, Berlin, 1991.

[2] M. Ishii, T. Sekiya, and T. Tomiyama. A very large-scale knowledge base for the knowledge intensive engineering framework. In *KB&KS'95, the Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, pp. 123–131, 1995.

[3] T. Tomiyama, D. Xue, Y. Umeda, H. Takeda, T. Kiriyama, and H. Yoshikawa. Systematizing design knowledge for intelligent CAD systems. In G.J. Olling and F. Kimura, editors, *Human Aspects in Computer Integrated Manufacturing, IFIP Transactions B-3*. North-Holland, Amsterdam, 1992.

[4] T. Tomiyama, Y. Umeda, M.Ishii, M. Yoshioka, and T. Kiriyama. Knowledge systematization for a knowledge intensive engineering framework. In T. Tomiyama, M. Mäntylä, and S. Finger, editors, *Knowledge Intensive CAD-1, Preprints of the first IFIP WG 5.2 Workshop on Knowledge Intensive CAD-1*, pp. 33–52. Chapman & Hall, 1996.

[5] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue. Metamodel: A key to intelligent CAD systems. *Research in Engineering Design*, Vol. 1, No. 1, pp. 19–34, 1989.

[6] T. Kiriyama, T. Tomiyama, and H. Yoshikawa. The use of qualitative physics for integrated design object modeling. In *Design Theory and Methodology (DTM '91)*, pp. 53–60. The American Society of Mechanical Engineers (ASME), New York, 1991.

[7] H. Takeda, T. Tomiyama, and H. Yoshikawa. A logical and computable framework for reasoning in design. In *Design Theory and Methodology (DTM '92)*. The American Society of Mechanical Engineers (ASME), 1992.

[8] ANSI/US PRO/IPO 100-1996. *Initial Graphics Exchange Specification IGES 5.3*. 1996.

[9] ISO TC184/SC4. *ISO 10303-1 Industrial Automation Systems and Integration - Product Data Representation and Exchange-, Part1: Overview and Fundamental Principles*. 1994.

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

[11] Y. Umeda, M. Ishii, M. Yoshioka, and T. Tomiyama. Supporting conceptual design based on the Function-Behavior-State modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, Vol. 10, No. 4, pp. 275–288, September 1996.

[12] H. Takeda, S. Hamada, T. Tomiyama, and H. Yoshikawa. A cognitive approach of the analysis of design processes. In *The Second ASME Design Theory and Methodology Conference*, pp. 153–160. The American Society of Mechanical Engineers (ASME), 1990.

[13] C. Hartshorne and P. Weiss, editors. *Collected Papers of Charles Sanders Peirce, Volume I-VI*. Harvard University Press, Cambridge, MA, 1931-1935.

[14] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, Vol. 13, pp. 27–39, 1980.

[15] S. Kondoh, R. K. Sato, and T. Tomiyama. Self-organization of the cellular manufacturing system. In H. van Brussel, J.-P. Kruth, and B. Lauwers, editors, *Proceedings of the 32nd CIRP International Seminar on Manufacturing Systems*, pp. 595–603, 1999.

[16] Y. Iwasaki, A. Farquhar, R. Fikes, and J. Rice. A web-based compositional modelng system for sharing of physical knowledge. In *Fifteenth International Joint Conference on Aritficial Intelligence*, pp. 494–450, Nagoya, Japan, 1997.

[17] M.R. Genesereth. Knowledge interchange format. In James Allen, Richard Fikes, and Erik Sandwall, editors, *Proceedings of the Conference of the Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, 1992.

[18] T. R. Gruber. Ontolingua: A mechanism to support portable ontlogies. Technical report KSL91-66, Knowledge Systems Laboratory, Stanford University, Stanford, 1992.

[19] B. Falkenhainer, A. Farquhar, D. Bobrow, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers. CML: A compositional modeling language. Technical report KSL94-16, Knowledge Systems Laboratory, Stanford University, Stanford, 1994.

[20] D. Xue, H. Takeda, T. Kiriyama, T. Tomiyama, and H. Yoshikawa. An intelligent integrated interactive CAD-a preliminary report. In D. Brown, M.B. Waldron, and H. Yoshikawa, editors, *Intelligent Computer Aided Design, IFIP Transactions B-4*. North-Holland, Amsterdam, 1992.

[21] P. P. Nayak and L. Joskowicz. Efficient compositional modeling for generating causal explanations. *Artificial Intelligence*, Vol. 83, No. 2, pp. 193–227, June 1996.