Res Eng Des (1989) 1:19–34

Research in
Engineering
Design

Metamodel: A Key to Intelligent CAD Systems

Tetsuo Tomiyama,* Takashia Kiriyama, Hideaki Takeda, Deye Xue, Hiroyuki Yoshikawa

Department of Precision Machinery Engineering, Faculty of Engineering, The University of Tokyo, Tokyo, Japan

Abstract. We introduce the metamodel as a new modeling framework for design objects based on General Design Theory, a mathematical model of design. Using General Design Theory, the metamodel concept can serve three functions: (1) as a central modeling mechanism to integrate models, (2) as a mechanism for modeling physical phenomena, and (3) as a tool for describing evolving design objects. Modeling with multiple points of view is realized by representing physical phenomena that occur in the design object and by constructing models with knowledge of physics and design from the metamodel. We illustrate the first and second functions of metamodels with an example based on naive physics, and we illustrate the third function of the metamodel through design experiments. Finally, we present two systems to illustrate how the metamodel mechanism can be implemented.

1 Introduction

Computer-aided design (CAD) systems are crucial tools for designers in several engineering fields including mechanical, electrical, aeronautical, architectural, and chemical engineering. A designer can examine and manipulate products on a graphic display by drawing figures, querying databases, calculating, and repeatedly modifying ideas. As products become increasingly complicated and technology becomes increasingly advanced, the amount of engineering knowledge needed by a single designer is increasing to nearly unmanageable amounts. This justifies the introduction of CAD systems, but such systems are not without problems.

The emphasis on advances in computer graphics technology would lead one to believe that handling geometric information freely in the ultimate goal in developing CAD systems. This is correct when only physical form is required, but in most engineering fields physical form is not the only type of information that is considered. For instance, aircraft designers must work with computation of aircraft wing behaviors. This computation requires not only geometric information about the wings, but also such information as material properties, welding conditions of structural members, and possible flight conditions. The same geometric information about the wings will be used in aerodynamic computation, fatigue analysis, and eventually in machining the hardware. This illustrates that engineering design requires various types of models, and the same information may be used in different models.

Thus, CAD systems must be equipped with facilities expressive enough to incorporate several kinds of information, including geometric information, in an integrated manner. At the same time, CAD systems must allow the designer flexibility in manipulating design knowledge, so that any particular chunk of knowledge can be used anywhere in a design process without transformation or translation.

CAD systems are in a state of transition between the traditional drawing systems and the new engineering systems in which geometric information is not the sole agent of design activity [ten Hagen and Tomiyama 1987]. These new systems are often called computer-aided engineering (CAE) systems. For instance, a two-dimensional CAD system can only generate drawings which are projections of real entities. Therefore, we foresee the need for three-dimensional modeling systems that have the ability to handle additional technological information.

A technique called product modeling is an example of current CAD technology and is applicable to such fields as robotics and dimensioning and tolerances. However, product modeling is not the final CAD solution because there is no end to the search for the ultimate datamodel. For instance, if we need to apply our three-dimensional solid modeler to robotics, e.g., simulation of robot arms, we develop a

* *Reprint requests:* Department of Precision Machinery Engineering, Faculty of Engineering, The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan

module that computes the arm movement. If we need to take tolerances into consideration, we must modify part of the system code and develop an additional package for tolerances on top of the nominal dimensions [Kimura, Suzuki, and Wingard 1986]. It is possible to add any scheme for a new application, but this manner of proceeding has the following drawbacks:

- We must repeatedly implement new packages because we have yet to obtain a universal scheme that can be applied to any engineering application.
- Schemes added on top of geometric modeling are *ad hoc* in one way or another. This prevents reuse of data in other applications and causes problems in maintenance and consistency among different data schemes.
- Each time we add a new scheme there may be slight changes to the system, yielding problems for system maintenance.

Therefore, CAD researchers have begun to stress the importance of seeking an ultimate, general purpose data scheme that goes beyond product modeling [ten Hagen and Tomiyama 1987; Gero 1987]. Until now there have been few reported achievements. A few results can be summarized as follows [Tomiyama and ten Hagen 1987, 1987a; Veth 1987]:

- This problem has much in common with the problem of knowledge representation in artificial intellegence (AI).
- We must discover flexible, preconception-free, yet efficient knowledge representation schemes (data models in the terminology of databases).
- Although geometric modeling plays an important role in mechanical design, there is little hope that such an ultimate data scheme can be found using conventional geometric modeling techniques [Arbab 1987].
- The basic concepts that must be incorporated into such a scheme are entities, relationships among entities, attributes of entities, and various kinds of reasoning that appear in design processes.

Efforts to develop intelligent CAD systems [Gero 1985, 1987; Eder 1987; Sriram and Adey 1986, 1987, 1987a, 1987b; ten Hagen and Tomiyama 1987] must be understood not only in the context of expert systems but also in the context of seeking a multipurpose modeling scheme for integration.

In this paper, we pursue yet another type of ultimate modeling scheme. *Metamodel* serves as a central modeling basis of CAD and allows flexible transformations between various models used in design, and it supports integration of various models. In Section 2, we discuss aspects of metamodels in design theory, and we clarify fundamental terms and concepts about modeling in design. We also discuss and formalize three aspects of the metamodel: metamodels as a framework to integrate models, metamodels as a framework to model physical phenomena, and metamodels as a tool for designers to evolve ideas about design objects. Section 3 concentrates on modeling of physical phenomena. Section 4 describes how metamodels are used and represented in design processes from an experimental point of view. Finally, in Section 5, we show two experimental systems that we are currently developing as examples of a *metamodeling* system.

2 Theory of Metamodels

2.1 *Modeling in Design*

A *model* is a *theory-based* set of descriptions about the object world. A model has properties that represent the object world, and these properties are selected and abstracted from entities and phenomena in the object world. In this definition, *modeling* is a process in which observed facts are filtered by a *theory* to formulate a world which itself is complete in terms of the theory. Figure 1 depicts this idea about how models are constructed from theories.

For example, consider a mechanical system such as the one shown in Fig. 2. A boundry representation geometric model of this system would focus on the system's geometric properties such as surfaces, lines, curves, and points. The theory behind the boundary representation model is algebraic geometry, and the geometric entities are represented mathematically as algebraic equations. In the same

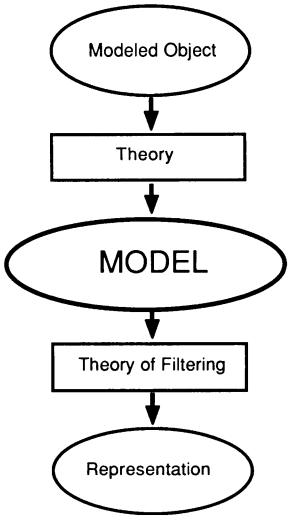

Fig. 1. Theories, models, and representations

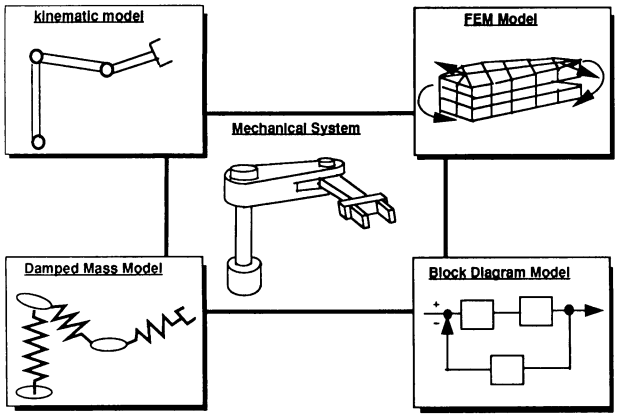

Fig. 2. Models for a mechanical system

way, a kinematic model is based on kinematics, a damped mass model is based on the mechanics of particles, a finite element model (FEM) is based on strength of materials, and a block diagram model for control is based on the classical control theory. However, two models can be transformed to each other only when the corresponding background theories are compatible. For example, it is impossible to transform an FEM model to a damped mass model without additional knowledge about the background theories. This suggests that, if we want to transform models to each other, we must make the corresponding background theories compatible.

Models and *representations* are different, too. For a given entity, as many models can exist as exist theories, and a single model can have many different representations. For instance, a geometric model can have wire-frame representations and surface representations for raster graphics both constructed from the identical model. Drawings are one kind of representation, mainly of geometric models. A technical document is another representation of a model.

Especially in mechanical design, we use various kinds of models, such as drawings, as the working tool to evaluate the functionalities of the design object. These include geometric models, kinematic models, dynamic models, mathematical models, strength models, and others. The problem is that the level and focus of the models are different. For example, the information used in manufacturing is too detailed for structural analyses. Therefore, different attributes sometimes might be considered to be the same, although their internal values may be completely inconsistent with each other. The metamodel mechanism solves this problem by unifying the different models to appear as a central design model. By having a central model, we can avoid the combinatorial explosion problem in implementing data transfer programs between models and maintaining consistency among them.

2.2 *Roles of Metamodels*

The metamodel concept has three aspects or roles. The first role of the metamodel is to serve as a central framework for integrating and unifying various models used in CAD systems. As discussed in the previous section, integration of models requires compatibility between the corresponding background theories. For instance, consider integrating the FEM model to evaluate strength with the dynamics model to evaluate dynamic behaviors as in Fig. 2. Obviously, the FEM model contains geometric information, material properties, and mesh information, whereas the dynamic model contains information about mass distribution in addition to geometrical information and material properties. Integration of these two models might be achieved by constructing a common database that contains geometrical information and material properties, and adding additional modules to generate meshes and compute mass distribution (Fig. 3). Note that mesh generation and computation of mass distribution are dependent on geometrical information and material properties, so changes made in geometrical properties should propagate to the mesh and mass distribution.

This method of integration is, accordingly, an *ad hoc* approach and has the following drawbacks:

- If we wish to integrate another model, it is necessary to modify the entire modeling scheme. The problem is that this modification is necessary every time we introduce a new model and that propagation of changes is not an easy task.
- The level and focus of models are different from each other. For example, mesh information can be generated from geometric information. However, this does not mean that the FEM model contains exactly the same external shape as the geometric model because minor features in the external shape are frequently omitted from struc-

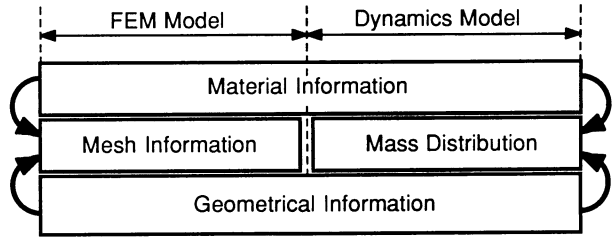

Fig. 3. Integrations of an FEM model and a dynamic model

tural computation. As stated previously, this problem arises from the incompatibility of the background theories.

The metamodel mechanism must adsorb incompatibility in the background theories. It should have not only the results of abstraction of attributes or properties from objects, but also the details of how the abstraction was made. In other words, the metamodel mechanism does not integrate models but instead integrates the background theories behind the models.

This discussion leads us to the second meaning of the metamodel; it is a mechanism to model physical phenomena. The metamodel mechanism should contain descriptions about how models are made. In Section 3, we will discuss this aspect of the metamodel in more detail.

Design is a process in which the designer builds ideas about artifacts to satisfy given specifications. Usually, the designer must build from imprecise, incomplete, and inconsistent specifications. Design objects evolve as the design process proceeds to arrive at the final state with precise, complete, and consistent descriptions. The third function of the metamodel mechanism is to serve as a work space for designers by providing them with various working models in the design process. We will examine this aspect in Section 4, after formalizing metamodels in the following sections.

2.3 *General Design Theory*

Design objects, such as a machine part or a floor plan, can be represented reasonably well now in computers through geometric modeling. Compared to design objects, design processes are not well described nor even understood. There are a number of efforts to create theories of design. For example, see [Hubka and Andreasen 1983; Hubka 1985; Eder 1987]. However, for the most part, these theories are not computer-implementable or computable theories [Kalay 1987]. Some are only collections of episodes or lessons obtained from practices over years. What is needed is a method to describe design processes *logically* so that we can trace design using computers [Veth 1987].

The metamodel mechanism was born from a computational approach to design processes. In this section, we introduce General Design Theory as the basis to formalize design processes and to describe design knowledge. (Readers are invited to refer to [Yoshikawa 1981, Tomiyama and Yoshikawa 1987] for more precise discussions.) In the next section we deal with the metamodel concept theoretically in the context of General Design Theory.

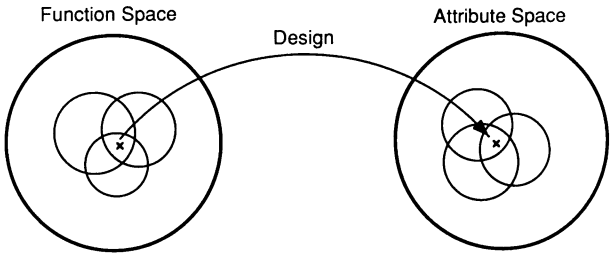

Fig. 4. Design process in the ideal knowledge

General Design Theory is based on axiomatic set theory; i.e., we begin with three axioms, the entity concept set, and its topology to describe design knowledge.

Axiom 1 (Axiom of recognition). Any entity can be recognized or described by attributes and/or other abstract concepts.

Axiom 2 (Axiom of correspondence). The entity set S' and the set of entity concepts (ideal) S have one-to-one correspondence.

Axiom 3 (Axiom of operation). The set of abstract concepts is a topology of the set of entity concepts.

We define design as a mapping from the attribute space to the function space, both of which are defined over the entity concept set. Here, we can introduce *ideal knowledge* that knows all of the elements of the entity set, and can describe each element without ambiguity by abstract concepts. The most significant result of having the ideal knowledge, which can be proven from the three axioms, is that design as a mapping from the function space to the attribute space immediately terminates when the specifications are described. (Since one knows everything perfectly in the ideal knowledge, when the specifications are completely described in terms of function, the solution is obtained in terms of attributes.) This says that design in the ideal knowledge is a mapping process from the function space to the attribute space and that there is no substantial computation required (Fig. 4). Thus, in the ideal knowledge we can immediately obtain design solutions.

Of course, this is not the case in the real design and we must take several characteristics into consideration. First, design is not a simple mapping process but a stepwise refinement process in which the designer seeks a solution that satisfies constraints. In addition, the concept of function is difficult to formalize objectively. It includes a sense of value that can be different from person to person.

Instead of using an objective definition, we use the concept of *behavior* to deal with function. Finally, the ideal knowledge does not take physical constraints into consideration and may produce design solutions such as perpetual machines.

These restrictions are considered in the *real knowledge* in which design is regarded as a process where the designer builds towards the design goal trying to satisfy specifications without violating physical constraints. In order to formalize the real knowledge, we first define a *physical law* as a description about the relationship between the physical quantities of entities and the field. The concept of physical laws is one of the abstract concepts formed when one looks at a physical phenomenon as manifestation of physical laws. Physical laws constrain entities in the real world; any feasible entity must be explicable by physical rules. This fact can be proven as a theorem.

Theorem 1. The set of physical law concepts is a base of the topology of the set of (feasible) entity concepts.

An interesting fact about the real knowledge is that we can prove finiteness or boundedness of our knowledge by having the following hypothesis:

Hypothesis. There exist finite subcoverings for any coverings of the set of feasible entity concepts made of sets chosen from the set of physical law concepts.

This hypothesis says that a feasible entity is explicable not by an infinite number but by a finite number (as small as possible) of physical laws. From this hypothesis, we can prove the following interesting theorems:

Theorem 2. In the real knowledge, there exists a distance between two different entities.

Theorem 3. In the real knowledge, it is possible to make a coverging subsequence from any design specifications and to find the design solution for the specifications.

Theorem 2 explains that every attribute has a value, if it is possible to measure the distance. Theorem 3 indicates that in the real knowledge a design process can be regarded as a convergence process, but solutions are not guaranteed to exist as a single point; it is also possible to obtain no solution or multiple solutions. However, it is guaranteed to find a solution by picking interesting points. This corresponds to the convergence of $\lim_{i\to\infty}(-1)^i$; if we choose only i which are odd numbers, this sequence converges to -1; if even numbers, we then get 1.

2.4 *Evolutionary Design Process Model*

Our next step is to formalize design processes in the real knowledge. For this purpose, we formally introduce the concept of *metamodels*, where a metamodel is a finite set of attributes and the metamodel set is the set of all metamodels.

Theorem 4. If we evolve a metamodel, we get an entity concept as the limit of evolution.

This theorem states that, if we describe solution candidates in attributes for the given specifications, by increasing the attributive descriptions we obtain an entity that is not always a solution. However, if we choose the set of physical law concepts as the metamodel set, the entity as the limit of evolution is the design solution.

Theorem 5. If we choose concepts that can be explained by such physical law concepts as the metamodel, we can describe the design specifications by the topology of metamodel, and there exists the design solution that is an element of this metamodel.

This theorem guarantees that we are able to design as far as specifications are given in terms of (physical) behaviors, and solutions are described in terms of attributes that can be measured by physical laws. Furthermore, solutions contain only objects that can be realized physically; in other words, we are not allowed to consider objects that contradict physical laws. At the same time, the theorem indicates a design process that is a stepwise, evolutionary transformation process with solutions that are obtained in a gradual refinement manner. Figure 5 depicts this design process based on the concept of metamodels incorporating physical laws, which we call the *evolutionary design process of metamodels*.

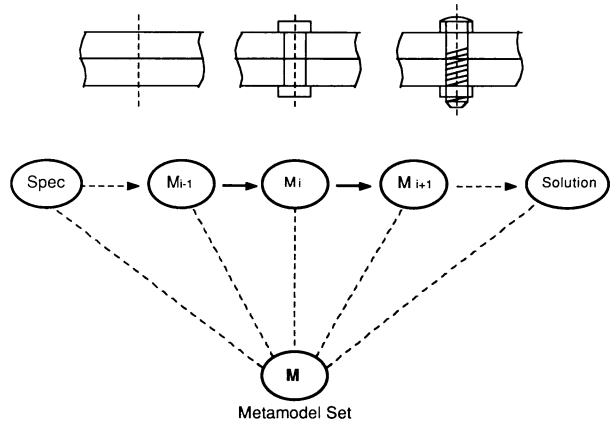

Fig. 5. Design process in the real knowledge

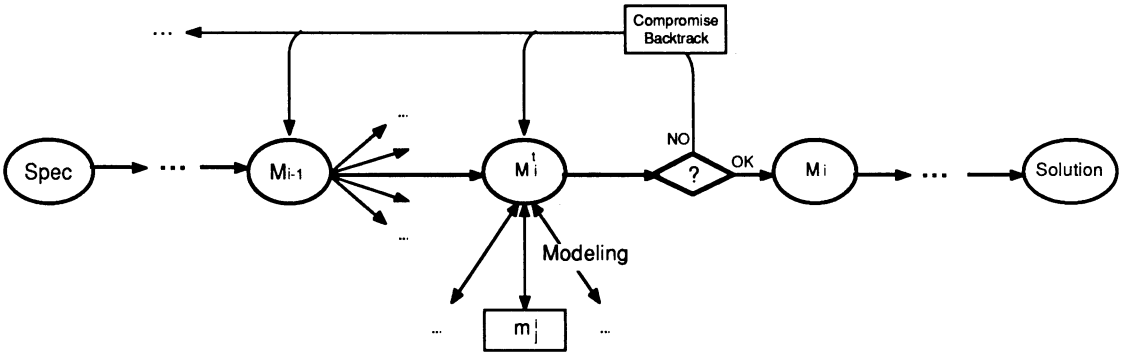

Fig. 6. Evolutionary design process. M: metamodel; M_i^t: temporary metamodel; M_j: model

In the ideal knowledge, design is a direct mapping process from the function space of the attribute space, while in the real knowledge, design is a stepwise, evolutionary transformation process. The designer will first generate candidate solutions for the given specifications. These candidates will be improved toward a final solution that satisfies the specifications. Figure 6 illustrates the evolutionary design process model in which the design object is stored in the central model, since models for various kinds of evaluation will be derived from this metamodel. The metamodel can be a set of logical assertions and a model will be constructed from it using knowledge for evaluation.

So far, we have formalized the three functions of metamodels using General Design Theory:

- a central modeling mechanism to integrate models,
- a mechanism to model physical phenomena (because metamodels must be explicable by physical laws), and
- a working tool for designers in an evolutionary design process model.

In the following sections, we will illustrate the implementation of these three modeling mechanisms in an intelligent CAD environment.

3 Modeling Physical Phenomena

This section deals with the first and second functions of metamodels; i.e., metamodels as a central modeling mechanism to integrate models and as a mechanism to model physical phenomena. As pointed out in Section 2.1, models are constructed by applying theories, so it is crucial to make background theories compatible rather than to make models compatible when models are integrated. Thus, we must represent information on physical phenomena that happen in design objects. This justifies the following attempt to represent and reason about physical phenomena.

3.1 Naive Physics

As an example, we will consider abrasion (Fig. 7). A model of abrasion consists of such information as the site where friction occurs, the forces on each part, and the material properties. A model of design objects cannot be constructed without knowing the theories of physical phenomena acting on the design object. We propose, as a mechanism to realize such a representation scheme, *naive physics* (or *qualitative physics*) [Bobrow 1984]. Our justification for this is that during conceptual design stages an object does not have complete attributes, therefore its behavior can be represented only qualitatively. However, we need knowledge about the phenomenon itself to construct models. The knowledge about the phenomenon for this example is listed below:

- friction is occurring on the two parts,
- force is being applied to the parts,
- thus abrasion is caused.

Qualitative physics reasons about physical phenomena to find out what effect that phenomena will have in the observed system. For example, consider the situation shown in Fig. 8 [Forbus 1984]. The situation consists of entities, namely the contained liquid and the boiler, and the fact that the boiler is heating the liquid. A physical law is then found to be applicable to this situation—that the heated water increases in temperature. The temperature rises and might reach the boiling point. Facts that can be derived in this process are:

- objects exist here,
- water temperature increases,

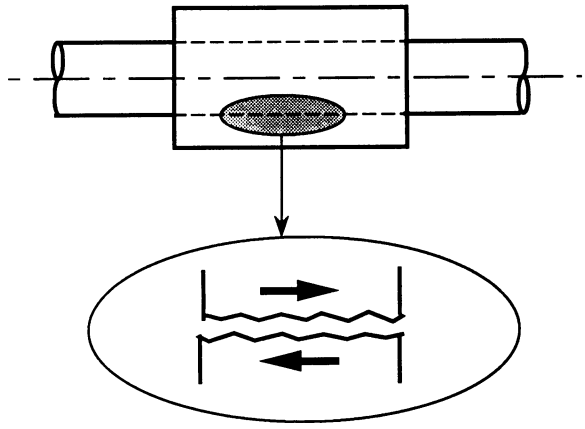

Fig. 7. Abrasion

- two naive laws: heated water increases its temperature and boiling occurs in water at the boiling point.

We employ the basic idea of Qualitative Process Theory [Forbus 1984] to represent physical phenomena in the context of metamodels. The theory consists of three notions, i.e., individual view, process view, and history. An individual view is a category of existing entities, such as water, a container, and a heater. A process view is a category of physical occurrence influence on entities. Heating is represented with a process view, having influence on water to make its temperature increase. In mechanical systems, energy transformation, transmission, and vibration are members of this category. A history is represented by a sequence of individual views and activated process views. A history of heating, for example, begins with a process *heating* acting on water, which causes the water temperature to rise. If the temperature reaches the boiling point, the process *boiling* becomes part of the history. In the metamodel theory, a history represents sequential phenomena occurring on or to objects, together with their causal dependency.

Individual and process views are supported by the preconditions that justify their existence. In the example of heating water, the process *boiling* is supported by the process *heating*. When the process heating ceases, boiling cannot continue.

individual views
a-work, a-hand, an-arm, a-cam, a-gravity
processes
(t0,t1)force(upward, a-cam, an-arm)
(t0,t1)force(upward, an-arm, a-hand)
(t0,t1)force(upward, a-hand, a-work)
(t0,t1)force(downward, a-spring, an-arm)
(t0,t1)gravity(downward, a-cam)
(t0,t1)gravity(downward, an-arm)
(t0,t1)gravity(downward, a-hand)
(t0,t1)gravity(downward, a-work)
(t0,t1)move-by-cam-mechanism(upward, a-cam, an-arm)
(t0,t1)move-by-connection(upward, an-arm, a-hand)
(t0,t1)move-by-grip(upward, a-hand, work)
transition
WORK: place(w0) → place(w1)
HAND: place(h0) → place(h1)
ARM: place(a0) → place(a1)
CAM: place(c0) → place(c1)

Fig. 9. Descriptions for the pick and place mechanism

3.2 Examples

We have implemented a program to examine Qualitative Process Theory and a means to qualitatively represent the behavior of machines. Figures 9 and 10 show an example of a simple mechanism for a "pick and place" job. A work located at the *place(w0)* moves via the *place(w1)* to the *place(w2)*. A robot hand grips the work and moves along the path *place(h0)*, *place(h1)*, and *place(w3)*. The hand is connected to an arm, and as the arm is pushed up by the cam the hand follows. Figure 11 represents the phenomena happing from the time *t0* when the work is at the *place(w0)* to the time *t1* when the work is at the *place(w1)*.

In this situation, three processes are occurring.

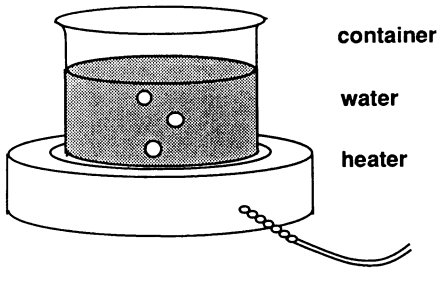

Fig. 8. Heating water

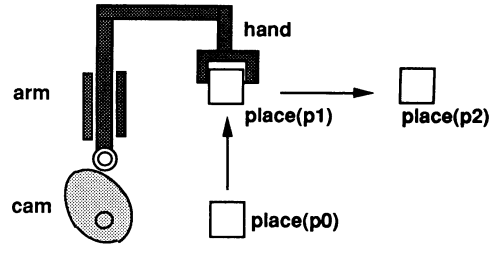

Fig. 10. Pick and place mechanism

```
%define {
((type (process move-by-cam-mechanism))
 (localname local-cam local-object local-direction)
 (conditions
   (direction local-direction)
   (can-move local-direction local-object)
   (contact local-direction local-cam local-object)
   (rotating local-cam))
 (add-conditions
   (apply-force local-direction local-cam local-object)
   (moving local-direction local-object))
 (relations
   (relation+
    (position local-direction local-object)
    (displacement local-direction local-cam)))))}
```

When
a cam is contacting with a follower in a direction,
and the cam is rotating,
and a follower can move in the direction,
then
the cam applies force to the follower in the direction,
and the follower is moving,
and position of follower is proportional to displacement of cam.

Fig. 11. Representation of phenomena

- The cam is pushing up the arm (process *move-by-cam-mechanism*)
- The arm is lifting the hand (process *move-by-connection*)
- The hand is lifting the work (process *move-by-grip*)

Process *move-by-cam-mechanism* is applicable to a pair consisting of a cam and an arm which is defined in the system as in Fig. 11.

Figure 12 is the result of a simulation, showing which individual views and processes occurred in this situation. The simulation begins with the given initial situation, namely the set of existing individual views and the process views, then instantiates possible individual views and process views. Processes make the value of variables either higher or lower, as shown in the result. We used this qualitative simulator to seek principles for representation of physical phenomena in the metamodel mechanism. Later in Chapter 5, we will describe another system that is based on these principles of qualitative simulation.

```
EPISODE 1
ACTIVE PROCESSES
(CAM-ROTATE+

(((PROCESS CAM-ROTATE))(LOCAL(LC-CAM A-CAM))))
(MOVE-BY-CAM-MECHANISM+
  (((PROCESS MOVE-BY-CAM-MECHANISM))
   (LOCAL(LC-CAM A-CAM)(LC-OBJECT AN-ARM)(LC-DIRECTION UPWARD))))
(MOVE-BY-FIX+
 (((PROCESS MOVE-BY-FIX))
  (LOCAL(LC-OBJECT1 AN-ARM)(LC-OBJECT2 A-HAND)
    (LC-DIRECTION UPWARD))))
(MOVE-BY-GRIP+
 (((PROCESS MOVE-BY-GRIP))
  (LOCAL(LC-HAND A-HAND)(LC-WORK A-WORK)(LC-DIRECTION UPWARD))))

eight more processes are omitted
ACTIVE INDIVIDUALS
(A-CAM (((IVIEW CAM)) (LOCAL)))
(AN-ARM (((IVIEW ARM)) (LOCAL)))
(A-HAND (((IVIEW HAND)) (LOCAL)))
(A-WORK (((IVIEW WORK)) (LOCAL)))
(A-SPRING (((IVIEW SPRING)) (LOCAL)))
(UPWARD (((IVIEW DIRECTION)) (LOCAL)))
(DOWNWARD (((IVIEW DIRECTION)) (LOCAL)))
(THE-GRAVITY-FIELD (((IVIEW GRAVITY-FIELD)) (LOCAL)))

CHANGING VARIABLES
(ANGLE A-CAM) INCREASING
(DISPLACEMENT UPWARD A-CAM) INCREASING
(POSITION UPWARD AN-ARM) INCREASING
(POSITION UPWARD A-HAND) INCREASING
(POSITION UPWARD A-WORK) INCREASING

POSSIBLE CHANGE OF ORDER
FROM: (ANGLE A-CAM) < (MAX-ANGLE A-CAM)
TO: (ANGLE A-CAM) = (MAX-ANGLE A-CAM)
```

Fig. 12. Result of simulation

4 Metamodel in the Evolutionary Design Process Model

4.1 Metamodels and Design Activities

We have discussed the first two functions of metamodels. In this section, we focus on the third function, i.e., metamodels in the evolutionary design process model. General Design Theory defines design as a mapping from the function space onto the attribute space, but in real design activities, design cannot be performed in such a straight-forward manner. There are many steps between the function space and the attribute space and the metamodel is used as a tool for designers to evolve design objects (in CAD terms as a working database). In this evolutionary design process model, design is treated as a sequence of unit processes that transfer one state of objects to another (Fig. 6).

Unit processes include the following:

(a) Designers observe the current metamodel and determine what is to be decided next.
(b) Designers think about and obtain solutions. In the narrow sense, this alone is regarded as a design process.
(c) Designers evaluate their solutions using various models. (These models are integrated by the first meaning of metamodel).
(d) Designers decide which solution to adopt and make the next metamodel. If there is no solution that satisfies the requirements, they must go back to and revise the current or older metamodel.

In order to perform the unit processes sequentially, metamodels must have information not only about design objects but also about processes, because metamodels define the boundaries between the unit processes. A unit process uses the metamodel as a precondition to make the next metamodel. Metamodels provide information for decision making and control. For example, a metamodel contains procedureal knowledge that specifies the design order.

4.2 Design Experiment

In this section, we verify and develop the evolutionary design process model in comparison to real design activities. We adopt the design experiment method [Yoshikawa, Arai, and Goto 1981] to recognize and analyze the design activities. The method consists of the following parts; we present one or more designers with a design problem and ask them to say what they think while designing. We record what they say, what they do, what they draw, and what they write until they complete their design. This recording and exploring is called protocol analysis [Ericsson and Simon 1980]. Drawings, sketches, and gestures, such as pointing to a part of a figure, play crucial roles in design. We record not only sayings but also figures and gestures. We reform them into procotol data after the completion of the design.

We have carried out several experiments with this method so far. The design problem we have selected is to design a part of the conveying mechanism for an automatic cigarette vending machine. The carrier is a motor-driven mechanism to take out one piece of goods from the stack. We made three pairs of subject designers composed from an engineer and five students, and presented each pair with the same problem to solve. It took three to four hours to solve the problem. Their design processes were recorded by VCR, and Fig. 13 shows an example of the protocol data.

We analyzed the protocol data in two ways, which turned out to be useful in recognizing how the concept of metamodels can be used in design activities. First, we picked up statements in which attributes of the object are represented explicitly, and reconstructed transitions of object descriptions (Fig. 14). Descriptions of an object are represented by boxes connected with full lines. A white arrow is a transition of an object description toward detailing. The black arrow indicates a transition of the designer's viewpoint. Transitions of object descriptions are associated with the evolution of the metamodel. Figure 14 indicates the evolution process of the metamodel. Some observations can be made:

- An evolution process is not linear, but has many branches. Even if a branch is selected, other branches are valid because they may be reused at later moments.
- Transitions of the designer's viewpoints are different from transitions of the metamodel, but they are closely related to each other.
- Because designers always have alternative solutions, they change their viewpoints from one to another when they are confronted with difficulties. A designer's thoughts are parallel and migrate frequently.
- Descriptions of objects are not the metamodel itself. There can be many viewpoints and the descriptions can be different from one viewpoint to another. However, a viewpoint and the descriptions associated with it are a possible representation of the attributes of objects.

Second, we analyzed the decision process. Designers proceed in the design by adding or revising their descriptions about objects. We defined a *decision*

" The problem is deciding how to take a package of cigarettes out. "
" It is not necessary to pick up a package with a hand-like mechanism."
" How about pushing it out with something like a forklift? "
" If the storage part is like the figure of the specifications, it is sufficient to push it out one at a time. "
" If it only has to drop a package out of the storage part, that is a good idea."
" If you don't like dropping, it can be pushed out and then carried by something like a conveyor belt. "
" If the storage part is like the figure, it is impossible to push out as in figure A(a)."
" It can be taken out from the bottom as in figure A(b). "
" The D parameter in figure A(b) must be greater than the thickness of the package. "
" OK. Then, let's think about an extruder. "
" How about a rubber conveyor belt (figure B)? "
" We can use the friction of rubber, but it is better to make raised sections on the rubber to move packages. "

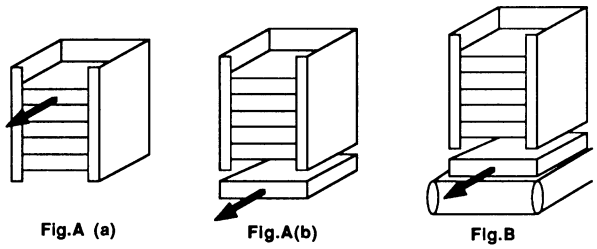

Fig. 13. Examples of protocol analysis

process as a transition from one metamodel to another, as represented by the white arrows in Fig. 14. Though a decision process looks complicated initially, we can identify five subprocesses that are rough breakdowns of decision processes. They include:

(1) *Awareness of the problem:* Determine what the problem to be solved is.
(2) *Suggestion:* Suggest key concepts necessary to solve the problem.
(3) *Development:* Identify candidates for solution of the problem from key concepts using various types of design knowledge.
(4) *Evaluation:* Evaluate the candidates in various ways, for example, structural computation, simulation of behaviors, cost evaluation, etc.
(5) *Decision:* Decide which candidate is adopted. Change the description of the object.

These five processes appear many times in design processes. We call this set of subprocesses a *design cycle*. Design processes can be constructed with a set of cycles. Figure 15 illustrates an example of this cycle found in the protocol data. This cycle is similar to the evolutionary process model. Each subprocess in this cycle corresponds to each unit process in the evolutionary process model, i.e., (1) to (a), (2) and (3) to (b), (4) to (c), and (5) to (d).

4.3 Logical Formalization of the Design Processes

In this section we discuss how the evolutionary design process model can be represented with a formalized model. Advantages of pursuing logical formalization are multifold. As shown in the next section, the results of this formalization can contribute to extracting constructs of a language that represents design knowledge which consists of knowledge about both design objects and processes. Furthermore, we can introduce various nonstandard logics to represent concepts that are particular to design [Veth 1987]. If we did this without logical formalization, the situation would be quite messy.

This approach for formalization is different from that of other research; see [Ullman, Dietterich, and Stauffer 1988]. They are more interested in extracting design operators, and seeing design as a sequential collection of operations, whereas our approach does not necessarily require such operators. We interpret a design process as a logical deductive process that can derive descriptions about the design objects from design knowledge and design specifications in the same way as a deductive process derives theorems from axioms.

Our approach makes clear that design has two

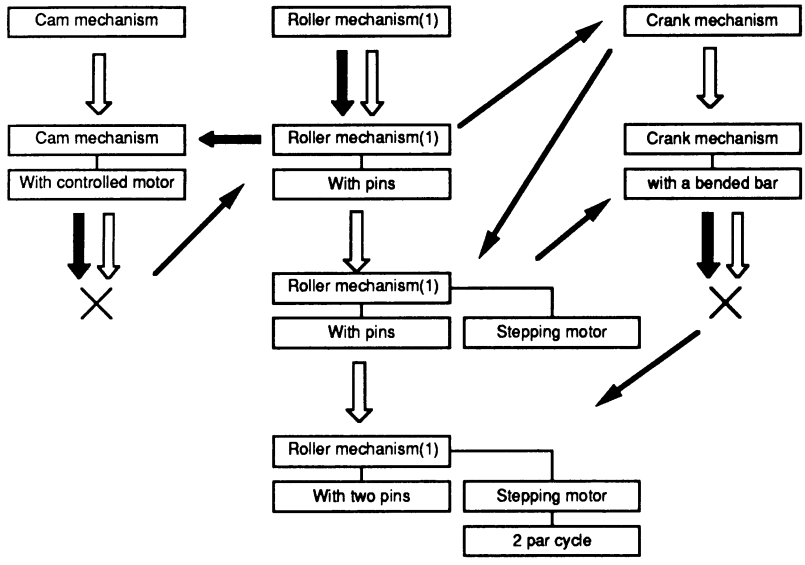

Fig. 14. Examples of transitions of object descriptions

"We must decide the mechanism to push out." [awareness of problem]
"How about the cam mechanism?" [suggestion]
"We must control the motor to stop each time the cam rotates once." [development]
"The cam mechanism makes shocks." [development]
"So, I am afraid that the machine's life will be short because of the shocks." [evaluation]
"We decide not to adapt the cam mechanism." [conclusion]

Fig. 15. Examples of design cycle

issues that must be considered. One is *parallelism* in design processes. Designers typically have some solutions that they later pursue, which can be done either successively, one after another, or quite randomly. The other is *retractability* of design processes. Designers decide some details at one point but may retract their decisions because they often proceed in their design processes by the trial-and-error method.

Although these two points are crucial in design processes, a simple deductive system does not have the capability to handle them. To overcome this problem, we introduce two non-standard logics. One is modal logic and the other is non-monotonic logic. Modal logic can be seen as the logic of necessity and possibility [Hughes and Cresswell 1972]. It is defined as an extension of normal proposition logic or predicate logic. It has two new symbols, **L** and **M**, which are called modal operators. For example, designers seem to use different kinds of truth values, as in "The length of this part must be 100 mm," or "The part A can be replaced by the part B." These statements can be naturally formalized by the following logical formulae:

$$\mathbf{L}eq(length(p),100),$$
$$\forall p\ \mathbf{M}\{p(A) \rightarrow p(B)\}.$$

The second formula suggests that higher order logic is also needed to describe reasonings in designing.

Modal logic is interpreted in multi-worlds, while the standard logic is interpreted in a single world. $\mathbf{L}p$ is valid in a certain world, if and only if predicate p is valid in every world which is accessible from that world. $\mathbf{M}p$ is valid in a certain world, if and only if predicate p is valid in one or more worlds which is accessible from that world. We use these properties to represent design processes, i.e., one world corresponds to one solution (or one proposal) and accessibility corresponds to the flow of design processes. In this interpretation $\mathbf{L}p$ represents what is always necessary through the entire design process, such as the required specifications. $\mathbf{M}p$ represents what becomes true during the design process, such as a particular property of a particular solution.

The other logic we use is non-monotonic logic [Reiter 1980]. it expresses reasoning with incomplete knowledge such as human thought. What people believe at a certain moment is based on what they known at that moment, so they revise or retract their belief if their knowledge changes. Such retractions cannot be expressed in standard logic. Non-monotonic logic has a new symbol **A** that designates belief which is believed if there is no con-

arise-of-problem(mechanism-to-check-soldout(main-body,?))
(world 1)**MA**mechanism-to-check-soldout(main-body, photo-sensor)
(world 1)**LA**high(cost(photo-sensor))
(world 2)**MA**mechanism-to-check-soldout(main-body, spring)
(world 2)mechanism-to-check-soldout(main-body, spring) -> be-stressed(package)
(world 2)**M**be-stressed(package)
(world 2)**L**(week(x) -> not be-stressed(x))
(world 2)**L**week(package)
(world 2)**L**(not be-stressed(x))
(world 2)**M**be-stressed(package) and L(not be-stressed(x)) is contradiction
(world 2)**M**(not mechanism-to-check-soldout(main-body, spring))
(world 1)**L**(not (high(cost(photo-sensor))))
evaluation
conclusion((mechanism-to-check-soldout(main-body,?)),?=photo-sensor))
L(mechanism-to-check-soldout(main-body, photo-sensor))

Fig. 16. Examples of logical formalization

trary fact. For example,

$$p \rightarrow \mathbf{A}q$$

means "if p is true and there is no contradiction to q, then q is true."

We use this type of non-monotonic expression to express weak statements in design processes that may be retracted in further processes. For example we represent statements of suggestion as non-monotonic expressions.

We have translated the protocol data into this logical form using modal and non-monotonic logics. Figure 16 shows examples of the logical representation. Although they are not strictly in logical form, they show how protocol data can be expressed. However, logic is generally an appropriate tool to formalize design processes, particularly to represent the evolutionary process model. We can conclude from this representation as follows:

- Most of the *development* subprocess and certain parts of the *suggestion* and *evaluation* subprocesses are represented as deductive processes.
- Modal expressions are useful to represent stepwise design processes and designers' viewpoints.
- Backtracking and avoidance of inconsistency in design processes are similar to those in non-monotonic reasoning.

5 Prototypes of Intelligent CAD Systems

In this section, we show two prototypes of intelligent CAD systems that illustrate the concept of metamodels; a language to describe design knowledge and a physical phenomenon modeler based on qualitative physics. Both systems are now under development in our laboratory.

5.1 *A Language for Intelligent CAD Systems*

Based on the concept of metamodels, a language that describes design knowledge of both design objects and processes has been developed in Smalltalk-80, an object-oriented langauge [Goldberg and Robson 1983]. This language is designed as an environment for easily implementing design knowledge and is called Integrated Data Description Language; (IDDL) originally in the Intelligent Integrated Interactive CAD (IIICAD) project conducted at the Centre for Mathematics and Computer Science in Amsterdam [Veth 1987].

IDDL is the kernel language of the IIICAD system. The configuration of IIICAD is illustrated in Fig. 17. System elements of IIICAD all use IDDL and, in particular, the supervisor controls the system using scenarios. In order to describe the stepwise refinement nature of design shown in Fig. 6, IDDL has a unit to represent the basic design cycle as discussed above:

- identify or generate a candidate solution for the given metamodel,
- make a model from the metamodel,
- evaluate the model, and
- modify or detail the metamodel so that it meets the specifications.

The language construct to do these in IDDL is a scenario. A scenario consists of a set of objects and logical rules over these objects. It represents a world in which the basic design cycle is completed.

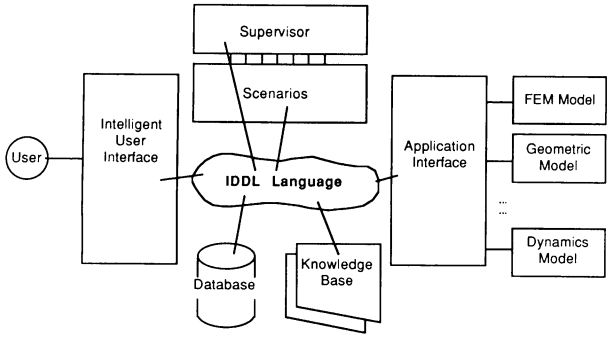

Fig. 17. Configuration of IIICAD

Semantically, a scenario represents a class of objects and its design. Thus, starting a scenario means starting a design and getting out of the scenario successfully means success of the design. When a scenario executes, it leaves objects in the database. Calling subscenarios means executing design subprocesses. A scenario corresponds also to a (modeling) view. By entering a scenario, it fetches a set of relevant objects and creates a world in which only relevant relationships among these objects can be seen.

In order to preserve or to see the solution of execution of scenarios easily, we use the concept of worlds. A world is a partition in the database, such that worlds are independent from each other, but can be linked so that changes in one world can propagate to others. There must be always at least one world that is active in the database. Scenarios are executed on the active world. A world consists of objects, facts, and attributes of the objects. IDDL employs modality (namely the possibility and necessity operators) and default reasoning in the form of the default operator. These operators are used to describe design processes more flexibly and to control executions of scenarios.

IDDL is, in principle, an object-oriented language in that objects send and receive messages to represent local, dynamic behaviors as function callings. The global, static behaviors of objects are described as relationships, i.e., as predicates. (In this sense IDDL is a combination between the object oriented and logic programming paradigms.) In the design process, we not only want to determine relations among objects, but we also want to directly operate on design objects. Functions are used for this purpose.

Figure 18 shows a screen from an experimental version of IDDL. We used Smalltalk-80 for the im-

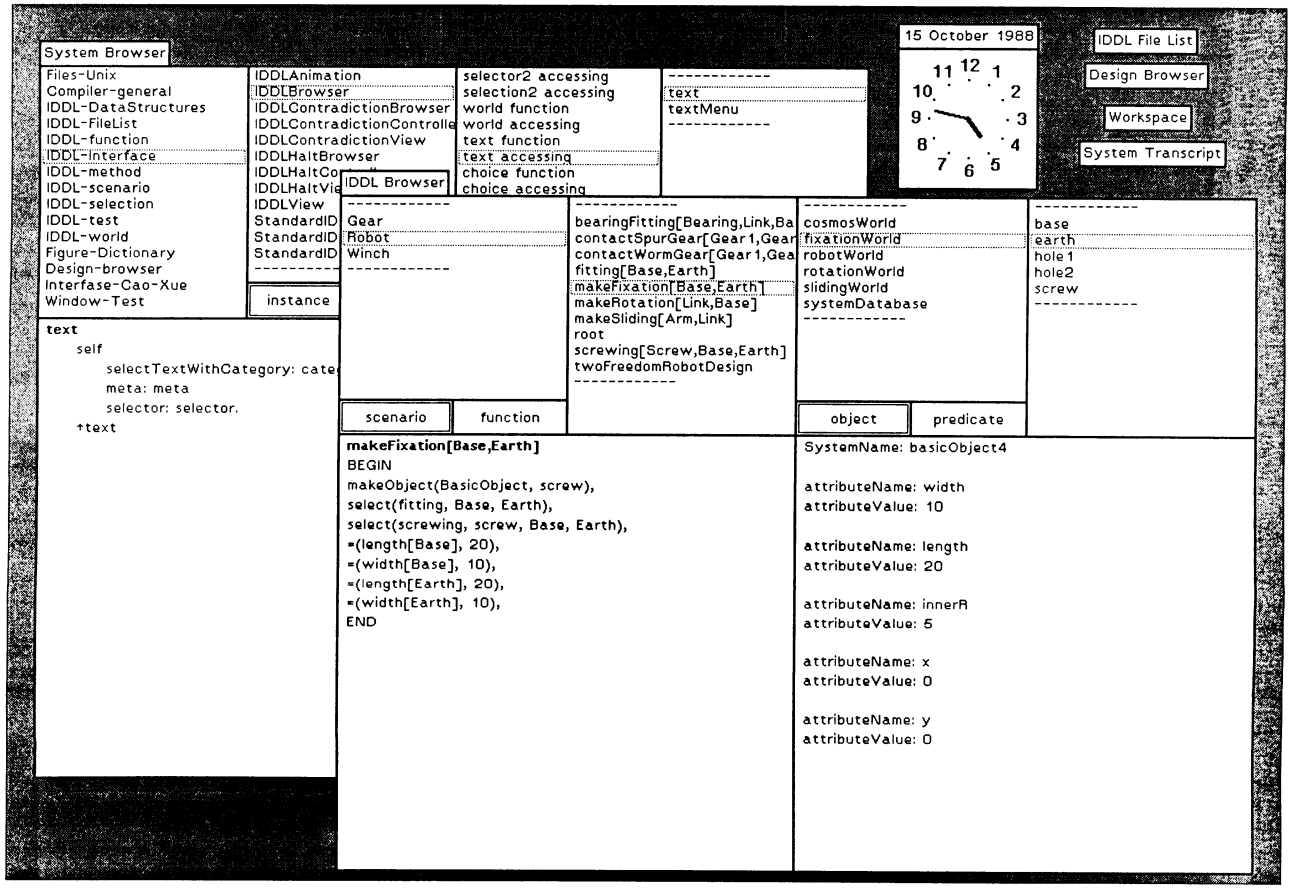

Fig. 18. IDDL browser

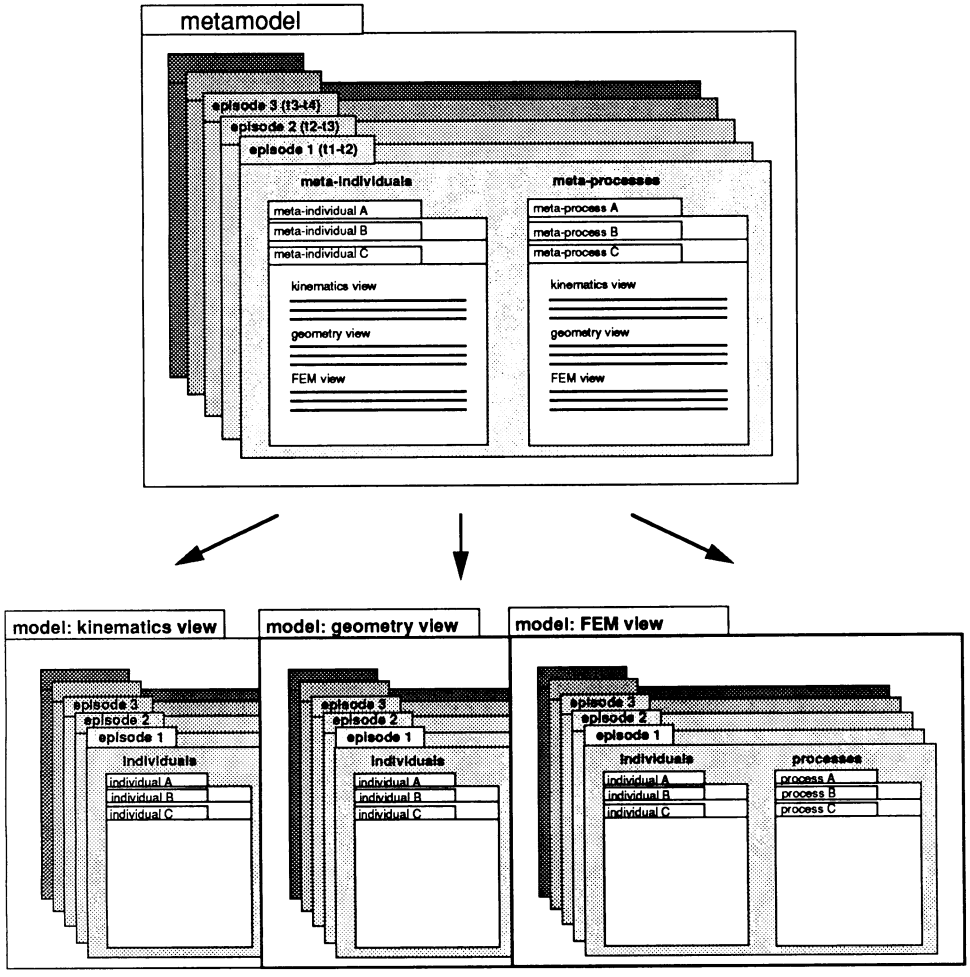

Fig. 19. Phenomenon modeler

plementation of IDDL, because objects in Smalltalk consist not only of procedures but also of data, so we can easily make design process and design objects. In addition, Smalltalk has an excellent user interface environment and interactive graphics environment.

5.2 *Phenomenon Modeler*

We implemented another system to illustrate our concepts of metamodels. The system intends to provide an environment in which qualitative representation is available to describe behaviors of design objects and models are constructed for particular viewpoints. Figure 19 shows the configuration of the system. The designer creates a metamodel in which he describes behaviors intended for the design object. Descriptions in a metamodel and models derived from the metamodel have causal dependency whose consistency is maintained by the system using a dependency network.

A metamodel consists of *meta-individuals* and *meta-processes*. A meta-individual is a collection of descriptions about existing entities; each description denotes a behavior seen from a particular viewpoint. Let us look back at the example in Section 3.2, that the cam is considered from the kinematics view as an object changing its height, although from the dynamics view it is an object applying force to its follower. In our metamodel theory, the cam is represented by a meta-individual which includes both the kinematics and dynamics aspects. Each of them describes how the cam behaves from the viewpoint concerned. A *meta-process* is also a collection of view-dependent effects on meta-individuals. Meta-individuals and meta-processes are interviewpoints descriptions of phenomena, hence they are on a meta level of individuals and processes.

Models are generated in each view with data derived from the metamodel. A *view* consists of relevant individuals and processes. A view gets relevant descriptions from the metamodel and puts all applicable local individuals and processes together to form a model. For example, suppose a cam is rotating and we are interested in the dynamics point of view (Fig. 20). Descriptions about dynamics, in-

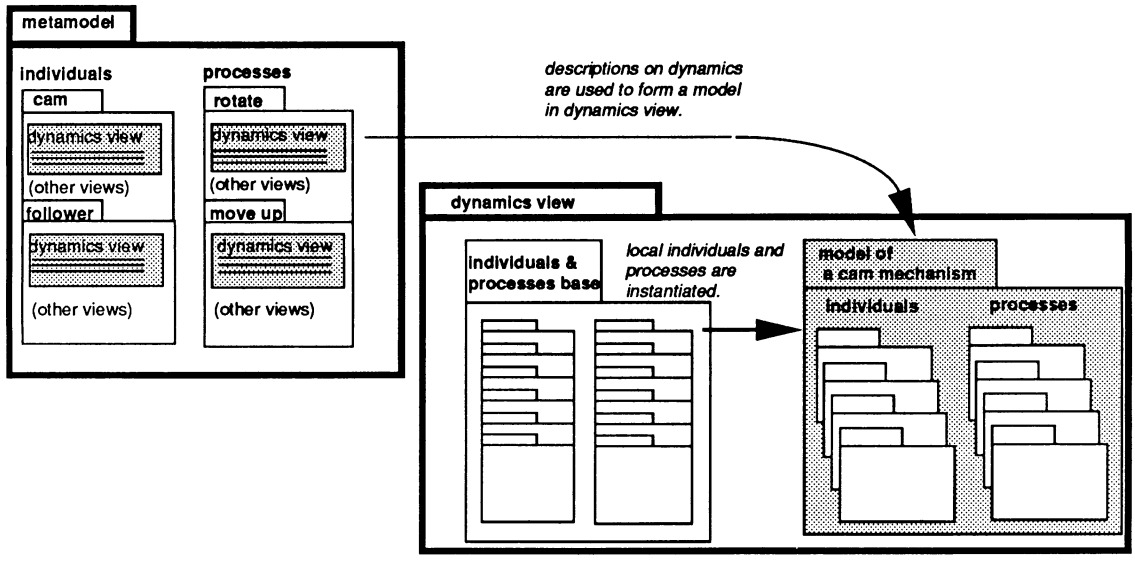

Fig. 20. A rotating cam in the metamodel

cluded in relevant meta-individuals and meta-processes, are sent to the dynamics view as soon as they appear in the metamodel. On the other hand, in the dynamics view, design knowledge (laws for a rotating cam) is instantiated, since it is found applicable to the current situation, and a dynamics model is composed from it.

This mechanism provides the functionality to generate an appropriate model from the metamodel at the time when the phenomenon the model represents as abstraction is recognized to occur. Therefore, there is no need for the designer to predict and prepare descriptions all about what is actually going to happen in a given situation. Rather, the system itself predicts with its knowledge phenomena and builds necessary models for evaluation.

The system is currently implemented in Smalltalk-80. It is a prototype of the metamodel mechanism and expected to become an object representation tool for our intelligent CAD system.

6 Conclusion

Designing is an intelligent activity of human beings, therefore tools for designing also must be intelligent to stimulate creativity and to serve as good assistants for designers. However, the simple question of how a designer actually thinks, decides, and innovates has yet to be understood. We believe that a good understanding about designing will result in useful tools for designers. Hence we started by formalizing designing, and then advanced into developing a CAD system guided by theories.

We introduced General Design Theory, a mathematical model of design. From the theory we deduced a new modeling framework for design objects called metamodel. According to the metamodel theory, a design object evolves over the design process, at the same time it is modeled from multiple points of view. This process, the evolutionary design process model, must be supported in future intelligent CAD environments. The metamodel concept has three functions—first as a central model to integrated models, second as a mechanism to model physical phenomena, and third as a tool to describe evolving design objects.

We illustrated our ideas to realize metamodels in the first and second functions based on naive physics. Modeling with multiple points of view is realized by representing physical phenomena that occur on the design object and by constructing models with knowledge in physics and design from the metamodel. Naive physics is suitable for representing qualitative behaviors and naive knowledge for this purpose.

Results of design experiments supports the third meaning of the metamodel. An evolution process can be represented by modal logic with two modal symbols denoting necessity and possibility respectively.

We presented two systems to see how the metamodel mechanism can be implemented. IDDL is a design language capable of representing knowledge about both design objects and processes in multiple worlds. The evolutionary design process model was represented in IDDL using scenarios. We have shown another system for designing qualitative be-

haviors and for deriving models using knowledge relevant to a particular viewpoint. This system demonstrated more directly how the metamodel mechanism can be realized.

References

[Arbab 1987]
Arbab, F.: "A paradigm for intelligent CAD," in [ten Hagen and Tomiyama 1987], pp. 20–39

[Bø, Estensen, and Warman 1986]
Bø, K., Estensen, L., and Warman, E.A. (eds.): *Proceedings of CAPE '86, Second International Conference on Computer Applications in Production and Engineering*, Copenhagen

[Bobrow 1984]
Bobrow, D.G. (ed.): *Qualitative Reasoning about Physical Systems*, The MIT Press, Cambridge, MA

[Eder 1987]
Eder, W.E. (ed.): *WDK 13, Proceedings of the 1987 International Conference on Engineering Design* (ICED 87), ASME, New York

[Ericsson and Simon 1980]
Ericsson, K.E. and Simon, H.A.: "Verbal reports as data," *Psychological Review*, 87(3), pp. 215–251

[Forbus 1984]
Forbus, K.D.: "Qualitative process theory," in [Bobrow 1984], pp. 85–168

[Gero 1985]
Gero, J.S. (ed.): *Knowledge Engineering in Computer Aided Design, Proceedings of the IFIP W.G. 5.2 Working Conference 1984 (Budapest, Hungary)*, North-Holland, Amsterdam

[Gero 1987]
Gero, J.S. (ed.): *Expert Systems in Computer Aided Design, Proceedings of the IFIP W.G. 5.2 Working Conference 1987 (Sydney, Australia)*, North-Holland, Amsterdam

[Gero 1988]
Gero, J.S. (ed.): *Artificial Intelligence in Engineering Design*, Elsevier, Amsterdam, Oxford, New York, Tokyo

[Goldberg and Robson 1983]
Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, MA, USA

[ten Hagen and Tomiyama 1987]
ten Hagen, P.J.W. and Tomiyama, T. (eds.): *Intelligent CAD Systems 1: Theoretical and Methodological Aspects*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo

[Hubka 1985]
Hubka, V. and the Programme Committee (eds.): *WDK 12, Proceedings of ICED 85 (Hamburg)-Theory and Practice of Engineering Design in International Comparison*, Heurista, Zurich

[Hubka and Andreasen 1983]
Hubka, V. and Andreasen, M.M. (eds.): *WDK 10, Proceedings of the International Conference on Engineering Design in Copenhagen 1983*, Heurista, Zurich

[Hughes and Cresswell 1972]
Hughes, G.E. and Cresswell, M.J.: *An Introduction to Modal Logic*, Methuen, London

[Kimura, Suzuki, and Wingard 1986]
Kimura, F., Suzuki, H., and Wingard, L.: "A uniform approach to dimensioning and tolerancing in product modelling," in [Bø, Estensen, and Warman 1986], pp. 165–171

[Kalay 1987]
Kalay, Y.E. (ed.): *Computability of Design*, John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore

[Reiter 1980]
Reiter, R: "A logic for default reasoning," *Artificial Intelligence*, 13, pp. 81–132

[Sata and Warman 1981]
Sata, T. and Warman, E.A. (eds.): *Man-Machine Communication in CAD/CAM, Proceedings of the IFIP WG5.2/5.3 Working Conference 1980 (Tokyo)*, North-Holland, Amsterdam

[Sriram and Adey 1986]
Sriram, D. and Adey, R. (eds.): *Applications of Artificial Intelligence in Engineering Problems, Proceedings of 1st International Conference 1986 (Southampton University, UK)*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo

[Sriram and Adey 1987]
Sriram, D. and Adey, R. (eds.): *Artificial Intelligence in Engineering: Tools and Techniques*, Computational Mechanics Publications, Southampton, UK

[Sriram and Adey 1987a]
Sriram, D. and Adey, R. (eds.): *Knowledge Based Expert Systems in Engineering: Planning and Design*, Computational Mechanics Publications, Southampton, UK

[Sriram and Adey 1987b]
Sriram, D. and Adey, R. (eds.): *Knowledge Based Expert Systems for Engineering: Classification, Education and Control*, Computational Mechanics Publications, Southampton, UK

[Tomiyama and ten Hagen 1987]
Tomiyama, T. and ten Hagen, P.J.W.: "The Concept of Intelligent Integrated Interactive CAD Systems," CWI Report No. CS-R8717, Centre for Mathematics and Computer Science, Amsterdam

[Tomiyama and ten Hagen 1987a]
Tomiyama, T. and ten Hagen, P.J.W.: "Representing Knowledge in Two Distinct Descriptions: Extensional vs. Intensional," CWI Report No. CS-R8718, Centre for Mathematics and Computer Science, Amsterdam

[Tomiyama and Yoshikawa 1987]
Tomiyama, T. and Yoshikawa, H.: "Extended general design theory," in [Yoshikawa and Warman 1987], pp. 95–130

[Ullman, Dietterich, and Stauffer 1988]
Ullman, D.G., Dietterich, T.G., and Stauffer, L.A.: "A model of the mechanical design process based on empirical data: a summary," in [Gero 1988], pp. 193–215

[Veth 1987]
Veth, B.: "An integrated data description language for coding design knowledge," in [ten Hagen and Tomiyama 1987], pp. 295–313

[Yosikawa 1981]
Yoshikawa, H.: "General design theory and a CAD system," in [Sata and Warman 1981], pp. 35–58

[Yoshikawa, Arai, and Goto 1981]
Yoshikawa, H., Arai, E., and Goto, T.: "Theory of design experiment," *Journal of JSPE* 47(7), pp. 830–835 (in Japanese)

[Yoshikawa and Warman 1987]
Yoshikawa, H. and Warman, E.A.: *Design Theory for CAD, Proceedings of the IFIP WG5.2 Working Conference 1985 (Tokyo)* North-Holland, Amsterdam.