Supporting Model Building Processes for Analysis with Prototypes

Takayuki Sekiya and **Tetsuo Tomiyama**
Research into Artifacts, Center for Engineering (RACE)
The University of Tokyo
Komaba 4-6-1, Meguro-ku, Tokyo 153-8904, JAPAN
e-mail: {sekiya,tomiyama}@race.u-tokyo.ac.jp

Abstract

In engineering design, a variety of computational tools are utilized for analysis. Each tool has its own model of a design object, such as an FEA (Finite Element Analysis) model. To user these models better and to realize better desgn quality, a system computation should support designers to use the tools in an integrated manner. For this purpose, we are developing a system named Knowledge Intensive Engineering Framework (KIEF), with knowledge about physical world. KIEF offers prototypes of analysis model components. The prototypes help a designer to build a model by providing an ontology of a modeling theory that gives a clue to organize such models. This paper discusses issues about the formalization of modeling processes needed for systematizing the ontology. We also demonstrate how to build a beam model on the KIEF system.

Introduction

Designers' engineering activities include designing an artifact that satisfies requirements, and analyzing if it really does using a variety of tools. A good design solution is obtained only through balancing analysis results. Therefore, it is necessary to develop a method to support designers in using a variety of design tools in an integrated manner.

Recent most commercial CAE systems provide a method to transfer CAD data to numerical analysis tools such as FEA systems, based on product modeling technologies and STEP (Fowler 1995).

However, model construction is an intellectual process and requires the designer's appropriate judgment, including answering to questions like which part of a product must be modeled and which kind of conditions must be applied to the model. This signifies that supporting data exchange is insufficient to develop a system which integrates various design tools.

When building a model, the designer uses a wide variety of engineering knowledge, from common sense knowledge about the physical world to domain specific knowledge about how to use tools. The Knowledge Intensive Engineering Framework (KIEF) (Tomiyama, Kiriyama, & Umeda 1994; Tomiyama *et al.* 1996) is our attempt to provide designers with such modeling knowledge in an integrated manner. However, our current KIEF system does not have enough power to advise designers in answering questions in the abovementioned modeling process.

To solve this problem, we pursue systematization of knowledge needed for modeling. For this purpose, this paper investigates the possibility of formalizing modeling processes. In particular, we focus on how information about design objects is modeled and modified during modeling processes.

We consider that a model consists of model fragments like in compositional modeling approach (Falkenhainer & Forbus 1991), and a modeling process is iterations of mapping from a model composed of general concepts to one composed of concepts specific to a modeling theory.

The rest of the paper is organized as follows. First, we describe the KIEF system as a modeling support tool. Then we discuss requirements to support modeling processes with a computational tool, and how to formalize modeling knowledge and modeling processes. An example of a modeling process is also illustrated. Finally, we discuss the results including related work, and conclude the paper.

Modeling on KIEF

Knowledge Intensive Engineering Framework

KIEF integrates existing design tools and supports the designer's activities on the tools, such as model building, model-based reasoning, and model validation.

The main features of KIEF are a *pluggable metamodel mechanism* (Yoshioka *et al.* 1993) which is a mechanism for integrating design tools and a *Very Large-scale Knowledge Base* (VLKB) which supports the metamodel mechanism by supplying primitive knowledge about the physical world (Figure 1).

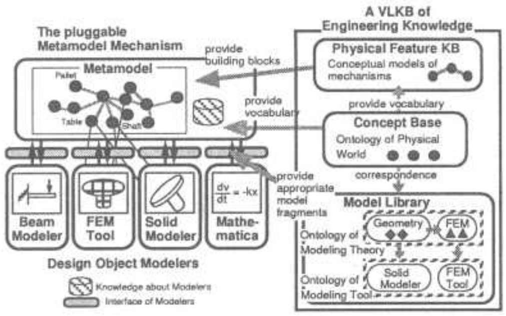


Figure 1: Knowledge Intensive Engineering Framework

The pluggable metamodel mechanism has symbolic representation of concepts about physical phenomena and mechanical components. A *metamodel* of a design object is represented as a network of relationships among concepts that appear in aspect models. Types of relationships include causal dependency among physical phenomena, arrangements of components, and quantitative relationships. An aspect model represents a model of the design object from a particular viewpoint and is usually dealt with by an existing external modeler. These concepts and relationships constitute the ontology of KIEF. The pluggable metamodel mechanism allows easily plugging in external modelers into KIEF.

VLKB for KIEF supplies fundamental knowledge about the physical world. VLKB consists of two primary knowledge bases, a *concept base* and *model libraries.* The concept base contains the fundamental and general ontology of KIEF, which is used for building a metamodel and representing engineering knowledge. The model libraries store ontologies specific to external modelers which correspond to general concepts in the concept base.

Modeling on KIEF

Roughly speaking, a model building process on KIEF has two steps. The first step is to extract a part of the conceptual network in the metamodel that is related to the tool the designer wants to use. The second step is to convert each concept to a tool-specific concept. We call the network of concepts extracted in the first step an *aspect model.* The third step is to add quantitative information to the aspect model, and then to prepare data for further numerical analysis.

Through these modeling steps, the designer makes decisions for modeling, while the system just suggests possibilities (such as tool-specific concepts) to him/her by automatically retrieving data from appropriate tools and calculates the values of data after the designer specifies necessary data.

Modeling Process for Analysis

Model of Modeling Process

Before we propose a model of modeling processes, we review two models of modeling processes previously investigated.

Tomiyama proposed synthesis/analysis oriented thought process models (Tomiyama *et al.* 1997). Both synthesis and analysis are basic thought processes repeatedly observed during a whole design process and complementary to each other. A model building process can be regarded as a synthesis thought process.

Chouiery proposed a practical framework for characterizing, evaluating, and selecting reformulation techniques, for reasoning about physical systems, which can be regarded as a general model of automated modeling (Choueiry *et al.* 1998). According to their work, there exists a process (called *model building*) to generate a model which contains knowledge of the physical structure as well as knowledge of the relevant physical phenomena.

Comparison of both Tomiyama's two models and Choueiry's model points out the following points to be considered in formalizing modeling processes.

- Knowledge about a modeling theory.
- Assembling necessary knowledge and information for modeling processes.

Taking these points into account, we propose a thought process model of modeling as follows.

1. Derivation of Physical Phenomena
Before starting a model building process, it is necessary to know what will occur to a modeling object.

2. Decision of the Modeling Task
The modeling goal is determined. The goal includes what to be solved by building the model and what to be paid attention to.

3. Selection of a Modeling Theory and a Tool
Next, we select an appropriate modeling theory to arrive at the modeling goal. Then, an appropriate computational tool is selected.

4. Setting up of Modeling Conditions
According to the selected modeling theory, we determine modeling conditions such as initial conditions and boundary conditions.

5. Modeling
This process consists of two subprocesses. One extracts information related to the selected modeling theory. The other translates them into concepts specific to the theory.

6. Data Construction on a Tool
To prepare data for the modeling tool, quantitative information, such as numerical value, is retrieved and added to the generated model.

7. Analysis and Evaluation
Finally, the model is analyzed and analysis results are evaluated with the tool.

Structure of Problems

We learn mathematics, physics, chemistry, etc, often by solving typical examples in textbooks. Similarly, in

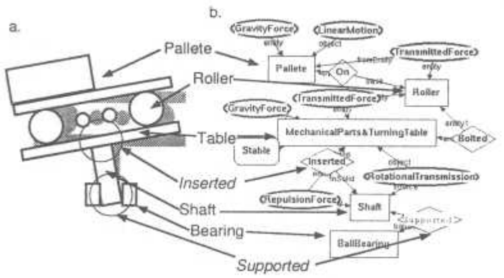


Figure 2: Cellular Automatic Warehouse

model building processes, the designer refers to existing typical models and cases.

However during design processes, some information of a design object has not yet been determined. Therefore, model building is difficult even if there is enough fragmentary knowledge about model components. t is more synthetical process to build a model based on such knowledge. Therefore, we have to develop methods to arrange typical models (called *prototypes*), to find an appropriate prototype from a knowledge base, and to use them in building a model for analysis. This is similar to case based reasoning (CBR), and we will compare our method with CBR in a later section.

Formalization of Modeling Knowledge and Modeling Process

Fundamental Ontology

In this paper, we illustrate an example of building a beam model of the turntable mechanism of a Cellular Automatic Warehouse (Sakao *et al.* 1996). Figure 2a shows the turntable mechanism, and Figure 2b depicts its topological structure consisting of fundamental ontology in KIEF.

First, we formalize fundamental ontology O_{real} and a model M. O_{real} is the fundamental ontology about the physical world, and is independent of modeling theories. O_{real} consists ontology of the following six kinds of concepts:

- "Entity"O_{en}: An entity represents an atomic, physical object. In the beam examples, $O_{en} = \{Shaft, Roller, TurningTable, ..\}$
- "Relation" O_{re}: A relation represents a relationship among entities. $O_{re} = \{Bolted, Inserted, Supported\}$
- "Attribute"O_{at}: An attribute is a concept attached to entities and takes a value to indicate the state of entities. $O_{at} = \{Length, Position, Deformation..\}$
- "Physical property" O_{pp}: A physical property is a concept that describes generic characteristic of entities. $O_{pp} = \{Stable, Strong..\}$
- "Physical phenomenon" O_{ph}: A physical phenomenon designates physical laws or rules that govern behaviors. $O_{ph} = \{BendingDeformation, Force, ..\}$
- "Physical rule" O_{pr}: A physical laws. $O_{pr} = \{Hooke'sLaw, SecondMotionLaw..\}$

$$O_{real} = O_{en} \cup O_{re} \cup O_{at} \cup O_{ph} \cup O_{pr} \cup O_{pp} \quad (1)$$

Structure predicate set S defines relationships among elements, such as $OccurTo(ph, e_1, e_2, ...)$ which means that physical phenomenon ph occurs to entities $e_1, e_2, ...$, and $HasRelation(r, e_1, e_2, ...)$ which means that there is a relationship r among entities $e_1, e_2, ...$

M consists of model elements $E_M = E(M)$ and predicates $P_M = P(M)$ which define properties of elements and relationships among elements.

$$M = (E_M, P_M),\ E_M = \{x_1, ..\},\ P_M \subset O_{real} \cup S \quad (2)$$

For example, the turntable mechanism (Figure 2b) is expressed as follows:

$$\begin{aligned} P(M) = \{&Pallet(x_1), On(x_2), Roller(x_3), Bolted(x_4), \\ &TurningTable(x_5), HasRelation(x_2, x_1, x_3), \\ &HasRelation(x_4, x_3, x_5), Inserted(x_6), Shaft(x_7), \\ &HasRelation(x_6, x_5, x_7), Supported(x_8), \\ &BallBearing(x_9), HasRelation(x_8, x_7, x_9)\}, \\ E(M) = \{&x_1, x_2, ..., x_9\} \end{aligned} \quad (3)$$

We introduce a physical feature f that describes a physical situation, and the designer constructs a metamodel with physical features:

$$\begin{aligned} f &= (E_{cond} \cup E_{drv}, P_{cond} \cup P_{drv}) \quad (4) \\ P_{cond} &\subset (O_{en} \cup O_{re} \cup O_{ph} \cup \{HasRelation, OccurTo\}), \\ P_{drv} &\subset (O_{ph} \cup \{OccurTo, Causality\}) \end{aligned}$$

To build a model, a model operation H_n which adds/removes elements/predicates in a model, is repeatedly evaluated. The following formula depicts M_{n+1} is generated by applying H_n to M_n:

$$M_{n+1} = H_n(M_n, ..) \quad (5)$$

There may be other arguments in addition to M_n if they are required. For example, operation *AddE* requires elements.

Ontology on Modeling Theory

A modeling theory T is a systematized knowledge required for model building, model operation, model-based reasoning, model validation, etc. The ontology of modeling theory O^T_{model} is categorized into related ontology O^T_{rl}, available ontology O^T_{av}, and derivable ontology O^T_{dr}. We use the ontology of Beam Modeler (a simple tool we developed for analyzing strength of a beam based on strength of materials) to explain the theory (Table 1).

$$O^T_{model} = O^T_{rl} \cup O^T_{av} \cup O^T_{dr} \quad (6)$$

Related ontology O^T_{rl} consists of concepts relevant to building a model according to modeling theory T.

Table 1: Ontology of Beam Model

	O_{rl}	O_{av}	O_{dr}
$O_{attribute}$	—	Force, Distributed Area, Supported Point...	Deformation, Bending Moment Diagram..
O_{entity}	Entity	Beam	—
$O_{phenomenon}$	Force	Distributed Force, Concentrated Force	Bending Deformation
$O_{relation}$	Relation	Hinged Support, Fixed Support, Free Support	—

Derivable ontology, O^T_{dr} is a set of derivable concepts when adopting a modeling theory T to a model. Both O^T_{rl} and O^T_{dr} are defined as subsets of fundamental ontology O_{real}. In case of Beam Modeler, "Force" belongs to related ontology, and "Deformation" belongs to derivable ontology.

$$O^T_{dr}, O^T_{rl} \subset O_{real} \tag{7}$$

During a modeling process, O^T_{rl} is used to filter out information in the metamodel only relevant to the model of interest for T.

Available ontology O^T_{av} consists of concepts specific to T, and used for composing a model. A model building process maps concepts in O^T_{rl} into concepts in O^T_{av}. We suppose that for each model element o^T_{av} there exists at least one feature f^T_{av} which satisfies the following formula:

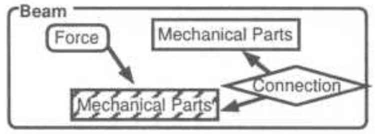
$$\forall o^T_{av} \in O^T_{av}, \exists f^T_{av}[(P_{cond}(f^T_{av}) \subset (O^T_{rl} \cup O^T_{av} \cup S)) \wedge (o^T_{av} \in P_{drv}(f^T_{av}))] \tag{8}$$

f^T_{av} defines o^T_{av} belonging to O^T_{av}, and defines a relationship between O_{real} and O^T_{av} with regard to T. When building a model, each part of the design object which has the same structure as f^T_{av} can be converted into o^T_{av}. The following formula depicts the operation FM to convert a fundamental concept into a modeling element. $AddE$ is a procedure to add elements to a model, and $AddP$ is a procedure to add predicates to a model.

$$\begin{aligned} M_{n+1} = FM(M_n, f^T_{av}) \quad \Leftrightarrow \\ \exists x_1, x_2.. \in E(M_n) \exists l_1, l_2.. \in P_{cond}(f^T_{av})[l_j(..x_i..) \\ \in P(M_n)] \to \forall l'_1, .. \in P_{drv}(f^T_{av})[M' = \\ AddE(M_n, x'_1, ..) \wedge M_{n+1} = AddP(M', l'_1, .)] \end{aligned} \tag{9}$$

The following set of formulae (10) describes an the example of a feature f^T_{av}, which is a definition of $o^T_{av} =$ "$Beam$," and Figure 3 depicts that graphical representation of f^T_{av}.

$$\begin{aligned} &E_{cond}(f^T_{av}) = \{x_1, x_2, x_3, x_4\} \\ &P_{cond}(f^T_{av}) = \{Force(x_1), MechanicalParts(x_2), \\ &\quad MechanicalParts(x_3), Connection(x_4), \\ &\quad OccurTo(x_1, x_2), HasRelation(x_4, x_2, x_1)\} \\ &E_{drv}(f^T_{av}) = \emptyset, P_{drv}(f^T_{av}) = \{Beam(x_2)\} \end{aligned} \tag{10}$$



The definition of **Beam**: This figure depicts that **Mechanical Parts** (the hatched rectangle node) can be regarded as a **Beam** if it has such a structure.

Figure 3: Definition of "Beam"

Prototype

Effective and efficient problem solving requires both knowledge about first principles and typical cases. In this paper, we call a typical analysis model, a *prototype*.

Formalization of Prototype A prototype is a pair of a conceptual network composed of model-specific ontology, and an analysis model. A conceptual network of a prototype M_{ty} in modeling theory T satisfies the following formulas:

$$\begin{aligned} M_{ty} &= E_{ty} \cup P_{ty} \\ P_{ty} &\subset O^T_{av} \cup S \\ \Phi_{ev}(M_{ty}) &\subset O^T_{dr} \end{aligned} \tag{11}$$

System of Prototypes A prototype is a similar concept to "case" of case-based reasoning (CBR). Prototypes should be systematically arranged like cases in CBR.

Generally speaking, it is not a simple task to categorize cases. It requires heuristics according to modeling theories, because originally the CBR technique is used for tackling ill-structured problems that have a large solution space. However, in building models often fundamental knowledge and model fragments are available from the modeling theory. Accordingly, we can use the ontology of a modeling theory to facilitate categorizing prototypes and to propose appropriate prototypes for a given problem by referring to the ontology.

In this research, we suppose that prototypes can be arranged hierarchically. Figure 4 depicts some functions to generate a prototype hierarchy.

In Figure 4, the function **add-new-prototype**, adds a new prototype to the existing prototype hierarchy(a list of prototypes *prototype-list*, a list of relationships among prototypes *link-list*). In Figure 4, when $M_{ty\,a}$ **is-part-of** $M_{ty\,b}$, in other words $PO(M_{ty\,a}, M_{ty\,b})$ holds, $M_{ty\,a}$ is obtained by mapping each element in $M_{ty\,a}$ to appropriate element in $M_{ty\,b}$, satisfies the formula:

$$P(M'_{ty\,a}) \subseteq P(M_{ty\,b}) \tag{12}$$

The generated hierarchy by this method will have a root prototype which consists of more abstract concepts, and that has simpler structure than any other prototypes. By referring to this hierarchy, it can be a simpler task to extract some prototypes which follow a certain condition (e.g. including a certain partial structure).

```
procedure add-new-prototype(new-M_ty, prototype-list , link-list )
        new-M_ty, existing-M_ty, generated-M_ty: prototype;
        prototype-list : A list of protoype;  link-list : A list of link between prototypes;
begin
        add-link-flag = false;
        foreach existing-M_ty in prototype-list do
        begin
                if is-super(new-M_ty ,existing-M_ty) then
                begin  register-new-link(new-M_ty , existing-M_ty ,link-list ); add-link-flag = true; end;
                if is-super(existing-M_ty ,new-M_ty ) then
                begin register-new-link(existing-M_ty ,new-M_ty , link-list ); add-link-flag = true; end;
        end;
        add new-M_ty to prototype-list ;
        if (add-link-flag == false) then
        begin
                generated-M_ty = generate-common-prototype(new-M_ty ,existing-M_ty);
                if not(generated-M_ty == nil) then  add-new-prototype(generated-M_ty, prototype-list , link-list );
        end;
end;
function is-super(M_ty1, M_ty2 )
        M_ty1,M_ty2 : prototype;  max-diff = 3 : number;
        /* A maximum number of different predicates/elements between models */
begin  return ( M_ty1  is-part-of  M_ty2 ) & (difference(M_ty1, M_ty2 ) <max-diff  ); end;
function  generate-common-prototype(M_ty1, M_ty2 )
        new-M_ty, M_ty1,M_ty2 : prototype;  max-diff = 3 : number;
begin
        new-M_ty = product(M_ty1, M_ty2 );
        if (difference(new-M_ty , M_ty1 ) > max-diff  ) | (difference(new-M_ty , M_ty2 ) > max-diff  )
                then return nil  else return new-M_ty ;
end;
```

Figure 4: Algorithm to Generate Hierarchy of Prototypes

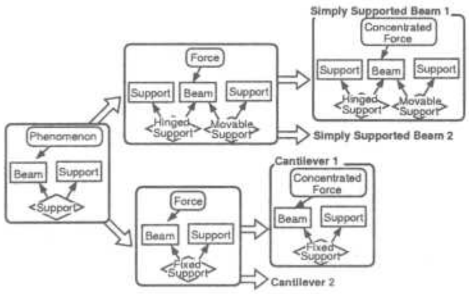


Figure 5: Part of Hierarchy of Prototypes

Figure 5 depicts the part of the hierarchy of prototypes obtained by applying the above-mentioned algorithm to prototypes in the "Beam Model." The root prototype consists of one beam supported at the one end, and that one phenomenon is applied to.

Modeling Processes

We here propose a formalized model of model building processes with regard to prototypes. In the following process model, we view modeling as mapping M into a target model M_m.

To explain our model of model building processes, first we built a model building mechanism based on KIEF according to the formalized modeling knowledge and modeling processes. Figure 6 depicts the system architecture.

1. Derivation of Physical Phenomenon

Before starting the model building process, the system reasons out what kind of phenomena will occur to the design object. As a result M_{drv} which contains M and the derived phenomena. A set of physical features, F_{drv} is used for deriving possible phenomena.

$$M_{drv} = PhenomenonDerivation(M, F_{drv}) \tag{13}$$

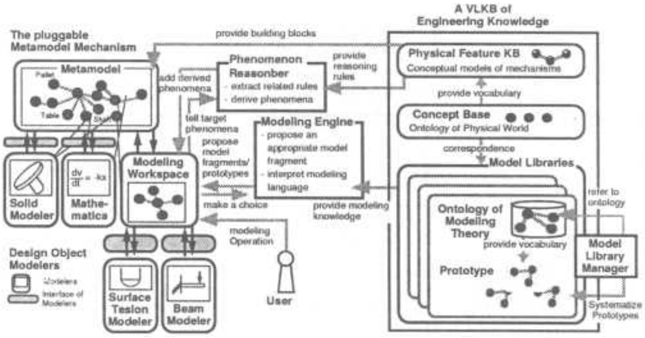


Figure 6: System Architecture

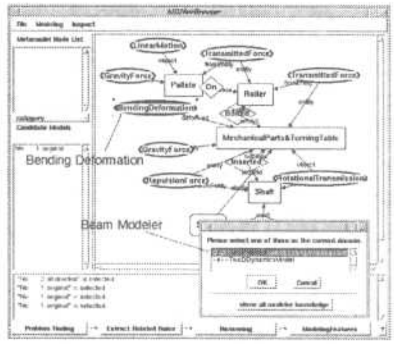


Figure 7: Selection of Modeling Theory

In this case, some phenomena such as "Deformation", "Gravity Force," "Transmitted Force," and "External Force"(represented as an oval node in Figure 2b) are derived:

$$\begin{aligned} P(M_{drv}) = P(M) \cup \{&TransmittedForce(x_{10}), \\ &OccurTo(x_{10}, x_3), BendingDeformation(x_{11}), \\ &OccurTo(x_{11}, x_4)...\} \\ E(M_{drv}) = E(M) \cup \{&x_{10}, ...\} \end{aligned} \tag{14}$$

2. Decision of a Modeling Goal

In our formalized modeling process model, a modeling goal R_{req} contains phenomena and attributes that can be reasoned and calculated with an analysis model. Therefore at this step the designer decides R_{req} which satisfies the following:

$$P(R_{req}) \subset (O_{at} \cup O_{ph}) \tag{15}$$

In this case, the designer selects "Bending Deformation", which is a hatched node shown in Figure7.

$$P(R_{req}) = \{BendingDeformation(x_{11})\} \tag{16}$$

3. Selection of the Modeling Theory and a Tool

At this step, the designer selects a theory T. According to T, the designer can compute a modeling goal R_{req}. The designer selects one of the theories, the derivable ontology of which, O^T_{dr} satisfies the following:

$$\forall l \in P(R_{req})[l \in O^T_{dr}] \tag{17}$$

T has a set of prototypes $\boldsymbol{M_{ty}} = \{M_{ty\,1}, M_{ty\,2}..\}$.

In this case, the system proposes "Beam Modeler" (a simple tool we developed for analyzing strength of a beam based on strength of materials) as an appropriate theory to analyze the selected modeling goals (see Figure 7) because the derivable ontology O_{dr} of "Beam Modeler" contains "Bending Deformation" (see Table 1). The designer selects "Beam Modeler" as T. Some prototypes of "Beam Modeler"are shown in Figure 4.

4. Setting Up Modeling Conditions

According to T, the designer sets up modeling conditions R_{assm}. There is the following relationship between R_{assm} and T:

$$P(R_{assm}) \subset (O^T_{rl} \cup O^T_{av} \cup S) \qquad (18)$$

At this step of modeling process, the system executes the operation *ApplyCondition* on M_{drv}, and finally M_{cond} is obtained.

$$\begin{aligned} M_{cnd} = ApplyCondition(M_{drv}, R_{assm}) \quad &\Leftrightarrow \\ \forall l_1, l_2 ... \in P(R_{assm})[\neg(l_i \in P(M_{drv})) &\rightarrow \\ M_{cnd} = AddPredicate(M_{drv}, l_1, l_2 ...)] & \qquad (19) \end{aligned}$$

In this case the conditions R_{assm} are *to regard table as "Beam"* and *to consider the "Beam" supported by "Fixed Support."*:

$$\begin{aligned} P(R_{assm}) &= \{FixedSupport(x_6), Beam(x_5)\} \\ P(M_{cnd}) &= P(M_{drv}) \cup P(R_{assm}) \qquad (20) \end{aligned}$$

5. Model Building

A set of prototypes $\boldsymbol{M_{ty}} = \{M_{ty\,1}, M_{ty\,2}..\}$ are hierarchically arranged by the algorithm Figure 4 depicts.

a. Selection of Prototype

At this step, the system reasons out a set of prototypes $\boldsymbol{M'_{ty}} (\subseteq \boldsymbol{M_{ty}})$. Each of the element in $\boldsymbol{M'_{ty}}$ satisfies modeling conditions R_{assm}. The phrase, "R_{assm} satisfies M_{ty}" means that $PO(R_{assm}, M_{ty})$ holds. The system calculates an additional conditions R'_{assm} for each prototype M'_{ty} ($\in \boldsymbol{M'_{ty}}$) to assist the designer to select a theory. First the system selects a prototype $M'_{ty\text{-}top}$ which satisfies the following:

$$\begin{aligned} &\forall M'_{ty} \in \boldsymbol{M'_{ty}} \exists M''_{ty\text{-}top} \in \boldsymbol{M_{ty}}[PO(M'_{ty\text{-}top}, M'_{ty}) \wedge \\ &\quad \neg(PO(M'_{ty\text{-}top}, M''_{ty\text{-}top}) \wedge PO(M''_{ty\text{-}top}, M'_{ty}))] \quad (21) \end{aligned}$$

Next assuming that $\boldsymbol{M'_{ty}}$ has hierarchically arrangement, the system generates additional assumptions R'_{assm} for each prototype M'_{ty}, that satisfies the following formula:

$$\begin{aligned} &\exists R_1, R_2 \in M'_{ty}[PO(R'_{assm}, M'_{ty}) \wedge \\ &\neg(PO(R_1, R'_{assm}) \wedge PO(R_1, M'_{ty\text{-}top})) \wedge \\ &\quad \neg(PO(R_2, R'_{assm}) \wedge PO(R_{min}, R_{assm}))] \qquad (22) \end{aligned}$$

The designer selects one prototype M'_{ty} by referring to R'_{assm}.

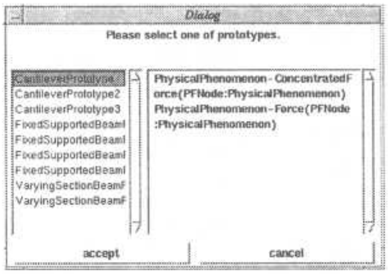


Figure 8: Presentation of Prototypes

In the Beam example, the system looks for prototypes which contains "Beam" and "FixedSupport." To help the designer choose an appropriate prototype, the system calculates additional conditions to distinguish the prototypes (see (21), (22)).

Figure 8 depicts the dialog window to select a prototype. The left part of the window shows that the prototype named "Cantilever" is selected as M'_{ty}, and the right part shows "Concentrated Force" which is the additional information R'_{assm} of "Cantilever":

$$\begin{aligned} P(M'_{ty}) = \{&FixedSupport(y_1), Beam(y_2), \\ &Support(y_3), ConcentratedForce(y_4), \\ &HasRelation(y_1, y_2, y_3), OccurTo(y_4, y_2)\} \\ P(R'_{assm}) = \{&ConcentratedForce(y_4)\} \qquad (23) \end{aligned}$$

b. Application of Prototype

The system applies M'_{ty} to M_{cond}. At this time, the system checks if M_{cond} has as substructure, a definition f_{av} of each model element $l \in M'_{ty}$:

$$\begin{aligned} &\forall f_1, f_2, .. \in F'_{av}[\bigwedge_{i=1}^{r}(FM(M_{cond}, f_i)) \neq M_{cond}) \rightarrow \\ &\qquad (M_{ab} = Abstraction(M_{cond}, f_1, f_2, ..., f_r))](24) \end{aligned}$$

In case there are some features with which the system cannot associate a model M_{cond} with, they are regarded as conditions required to use prototype M'_{ty}.

In this case, the designer selects "Cantilever" as M'_{ty} that consists of a cantilever to which a concentrated force is applied, because there is no part which supports the table, but some other prototypes requires that.

The system refers to the definition of each model element of which the prototype is composed (for example, 11), and translates a part of the design object into a model element. If the prototype can be generated, the system analyzes the design object with an equation set prepared for the selected prototype. In Figure 9, the hatched nodes means each of them is successfully translated into a model element the prototype contains.

6. Data Construction on a Tool

The system sets up all formula by referring to the prototype M'_{ty} when the system successfully applied M'_{ty} to the design object.

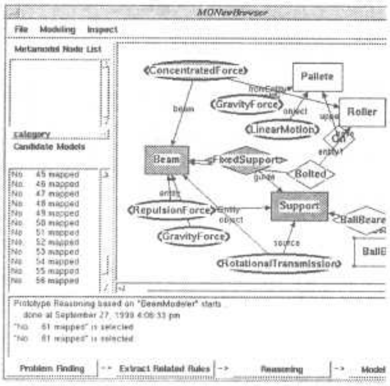


Figure 9: Application of Prototypes

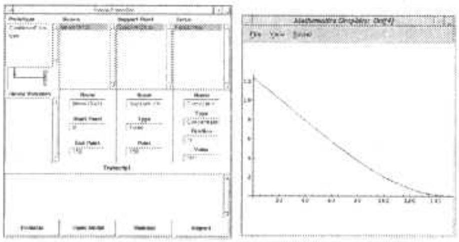


Figure 10: Beam Modeler

In case of Beam Modeler, quantitative data such as the length of the beam and Young's modulus, and an equation to calculate deflection of the beam are required for building an analysis model on the tool. They are prepared together with a prototype beforehand. KIEF supports assigning data to an analysis model by automatically searching a value of each quantitative data registered in KIEF.

7. Analysis and Evaluation

Finally, the system conducts the analysis to evaluate the model M. Figure 10 shows a screen hardcopy of Beam Modeler.

Discussions and Related Works

We proposed a framework that consists of prototypes and model-specific concepts to describe knowledge required for modeling. This framework improves reusability of knowledge, because the definition of a model-specific concept can be commonly used across several prototypes.

In this research, prototypes in a certain domain can be hierarchically arranged by referring to modeling ontology. Each prototype gives information that is necessary to build a model, and the prototype hierarchy supports designers to understand difference among prototypes.

However, currently there is no way to judge if a selected prototype is adequate for a given situation. We also have to develop algorithms to modify existing prototypes in order to deal with various design objects and situations.

Cases of CBR cannot be always arranged as elegantly as prototypes in our research, because cases of CBR are not based on well-structured ontologies. However, there is an algorithm to calculate how problems and cases resemble each other in CBR.

Our idea is influenced by Graph of Models (Addanki, Cremonini, & Penberthy 1991). However prototypes are hierarchically arranged automatically, while Graph of Models assumes a well-structured model set prepared beforehand. In our method, ontology should be prepared, but it provides more flexible prototype structure than Graph of Models.

In this research, each model-specific concept is described with generic ontology. These relationships among generic concepts and model-specific concepts can enrich generic ontology. There is no research to deal with ontology in this way.

While our method is most fundamental and allows domain independent modeling processes, domain dependent knowledge needs more careful treatment.

Conclusion

This paper formalized modeling processes and ontology required during model building processes. We introduced prototypes of model that represent typical fragmented knowledge for building models. According to the formalized ontology and modeling processes, we developed a modeling support system based on KIEF. Our method to support modeling can suggest to the designer modeling theories and prototypes that are appropriate for the modeling goal.

Future work includes the following:

- Improving a method to adopt prototypes to a design object (e.g. combining existing prototypes).
- Developing a mechanism to support building ontology.
- Collecting ontology in real design processes, including domain dependent ontology.

Acknowledgment

This research was funded by the "Modeling of Synthesis" project of the Japan Society of for the Promotion of Science's Research for the Future Program under the contract number, JSPS-RFTF 96P00701.

References

Addanki, S.; Cremonini, R.; and Penberthy, J. 1991. Graphs of models. *Artificial Intelligence* 51(1-3):145–177.

Choueiry, B. Y.; McIlraith, S.; Iwasaki, Y.; Loeser, T.; Neller, T.; Engelmore, R. S.; and Fikes, R. 1998. Thoughts towards a practical theory of reformulation for reasoning about physical systems. In *Working notes of the Symposium on Abstraction, Reformulation, and Approximation (SARA'98)*, 25–36. CA, USA: Pacific Grove.

Falkenhainer, B., and Forbus, K. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51(1-3):95–143.

Fowler, J. 1995. *STEP for Data Management, Exchange and Sharing.* Technology Appraisals Ltd, 1 edition.

Sakao, T.; Kondoh, S.; Umeda, Y.; and Tomiyama, T. 1996. The development of a cellular automatic warehouse. In *IEEE/RSJ International Conference on Intelligent Robots and Systems '96 (IROS '96)*, 324–331.

Tomiyama, T.; Umeda, Y.; Ishii, M.; and Yoshioka, M. 1996. Knowledge systematization for a knowledge intensive engineering framework. In Tomiyama, T.; Mäntylä, M.; and Finger, S., eds., *Knowledge Intensive CAD Volume 1*, 33–52. Chapman & Hall.

Tomiyama, T.; Murakami, T.; Washio, T.; Kubota, A.; Takeda, H.; Kiriyama, T.; Umeda, Y.; and Yoshioka, M. 1997. The modeling of synthesis – from the viewpoint of design knowledge. In Riitahuhta, A., ed., *Proceedings of the 11th International Conference on Engineering Design*, 97–100.

Tomiyama, T.; Kiriyama, T.; and Umeda, Y. 1994. Toward knowledge intensive engineering. In Fuchi, K., and Yokoi, T., eds., *Knowledge Building and Knowledge Sharing.* Tokyo, Osaka, and Kyoto: Ohmsha.

Yoshioka, M.; Nakamura, M.; Tomiyama, T.; and Yoshikawa, H. 1993. A design process model with multiple design object models. In *Design Theory and Methodology (DTM '93)*, 7–14. New York: ASME.