Pluggable Metamodel Mechanism: A Framework of an Integrated Design Object Modeling Environment

Masaharu Yoshioka
Research and Development Department
National Center for Science Information Systems
Ootsuka 3-29-1, Bunkyo-ku, Tokyo, 112, Japan
Tel: +81-3-3942-7003
Fax: +81-3-5395-7064
e-mail: yoshioka@rd.nacsis.ac.jp

Tetsuo Tomiyama
Department of Precision Machinery Engineering
Faculty of Engineering, The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113, Japan
Tel: +81-3-3812-2111 (ext. 6454)
Fax: +81-3-3812-8849
e-mail: tomiyama@zzz.pe.u-tokyo.ac.jp

Abstract

We have been developing a computational framework for knowledge intensive engineering called KIEF (Knowledge Intensive Engineering Framework). KIEF should have a mechanism for integrating and maintaining consistency among various models, such as a geometric model, kinematic model, and finite element model, that are required to conduct mechatronics design. Commercial design tools must be integrated into this mechanism. KIEF deals with these multiple design object models through the metamodel mechanism. A metamodel is a model that represents relationships among concepts used in these design object models. This paper proposes a pluggable metamodel mechanism that allows to plug in external design tools and modeling systems. It describes a prototype of the pluggable metamodel system and illustrates an example of ball screw design.

1. Introduction

Knowledge intensive engineering is a new style of engineering to assist engineering activities in various product life cycle stages based on intensive use of various kinds of engineering knowledge [Tomiyama94, Tomiyama96]. We have been developing a computational framework for knowledge intensive engineering called KIEF (Knowledge Intensive Engineering Framework). Since engineering activities in various product life cycle stages relates each other, KIEF should have a mechanism for integrating and maintaining consistency among various models. Speaking of mechatoronics design, we need to integrate aspect models such as a geometric model, kinematic model, finite element model dynamics model, and control diagram. In addition, since many commercial design tools are practically used in design processes, KIEF also requires to integrate these commercial tools.

For integrating these design object aspect models, we have formalized a concept called a metamodel mechanism [Kiriyama91]. The metamodel mechanism maintains relationships among concepts such as causal dependency, translation, and attribute-of used in these aspect models (Figure 1).

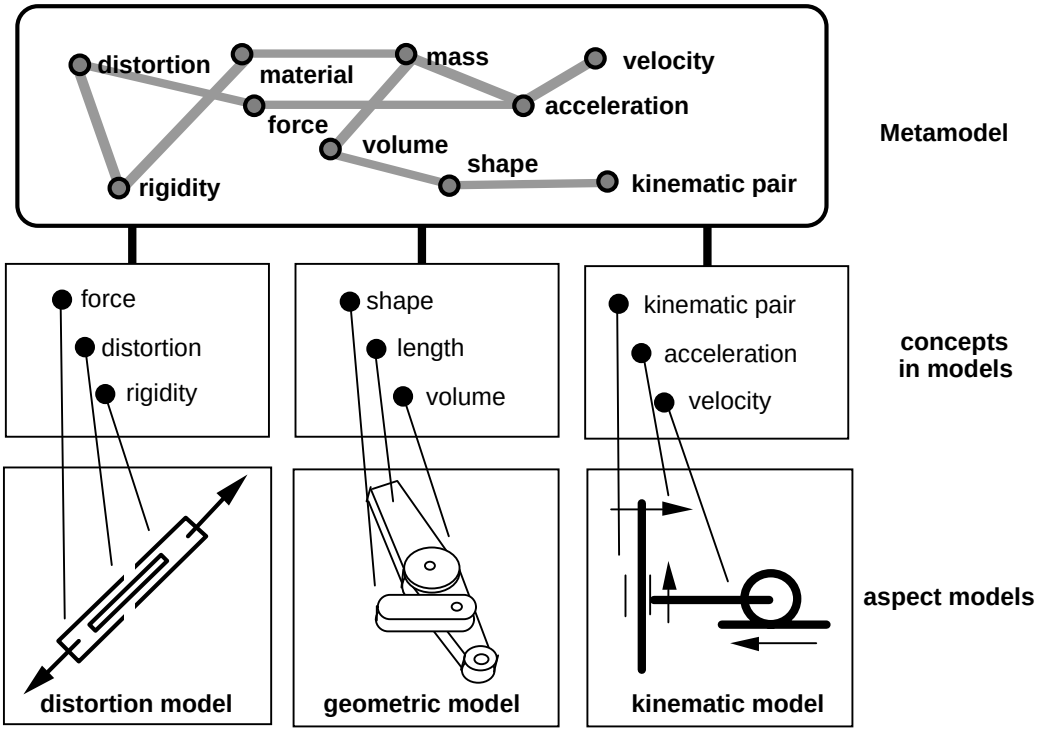


Figure 1 The metamodel mechanism

In addition, we have formalized a modeling process of a design object as operations to the metamodel to build aspect models [Yoshioka93]. An aspect model is a model of design object from a particular point of view called aspect, such as deformation, shape, and motion. We think there are two steps in building aspect models. In the first step, a designer builds a general qualitative model that represents relationships among concepts used to model the design object. In this step, the designer sometimes abstracts and simplifies the model. After building the qualitative model, the designer builds an aspect model by feeding data from existing models to a new model.

Integration of design object models has actively been studied. IGES (Initial Graphics Exchange Specification) aims to integrate graphical data among CAD. Since IGES supports only graphical data, it is not sufficient to handle all of data used in the whole design process. Thus started research on STEP (Standard for the Exchanging of Product model data) [ISO94]. STEP aims to integrate design object models at product data level in the EXPRESS language. However, STEP does not consider “ontological” level integration that the metamodel and the PACT project [Cutkosky93] address.

To incorporate various commercial design object modelers available, in this paper, we expand the framework of the metamodel mechanism to the Pluggable Metamodel Mechanism that allows to plug in existing design tools and modelers. We build a schema to describe what these tools and modelers can do and what kinds of data they require.

This paper is divided into five sections. Section 2 illustrates the framework of the metamodel and formalization of a modeling process based on this framework. Section 3 discusses the pluggable metamodel mechanism, a schema to describe plug-in tools, and methodology for transferring data among plug-in design object modelers. Section 4 demonstrates a prototype system of the pluggable metamodel mechanism. Section 5 compares our approach with other related work and Section 6 concludes the paper.

2. The Metamodel Mechanism

A design object can usually be represented with various aspect models that encompass a wide variety of different dimensions, i.e., domains of concepts for modeling, levels of abstraction, and levels of approximation. For instance, domains include function, geometry, and kinematic. A functional model represents the designer's intention, a geometric model represents shape, and a kinematic model represents motion of the design object. On the contrary, a beam model and a finite element model are used for evaluating strength at different levels of abstraction. A kinematic model with friction and without are used for evaluating motion at different levels of simplification. These aspect models are not independent of each other; therefore, we need to have a mechanism to maintain consistency among aspect models.

The metamodel mechanism is a modeling framework for integrating multiple aspect models [Tomiyama89, Kiriyama91]. It has symbolic representation of concepts about physical phenomena and mechanical components (Figure 1). A metamodel of a design object is represented as a network of relationships among concepts that appear in the aspect models. Types of relationship include causal dependency among physical phenomena, arrangements of components, and attribute relationships.

The designer builds an initial qualitative model of the principal physical behavior and the structure of the design object. We call this initial model a primary model. The primary model is an aspect model that represents the designer's mental model about behavior. An initial metamodel is generated from the primary model through qualitative reasoning.

An aspect model used by selected modeler is generated from the metamodel as follows.

1. Determine abstraction level for the selected aspect modeler (Figure 2-(a)).
   The metamodel mechanism determines the most appropriate abstraction level for the selected aspect modeler based on the knowledge about building aspect models. The metamodel mechanism suggests concepts included in the determined abstraction level and needed for the selected modeler, and the designer has to give abstract descriptions of the design object using these concepts. For example, consider the design of a robot arm. If the metamodel mechanism reasons out that distortion is possibly to happen, the metamodel mechanism suggests to describe the arm only using concepts such as beam, load, support, and bending. Then the designer abstracts the arm concept in the domain of geometry as a beam. This abstraction is computationally done by unification that is an operation to create a new instance that delegates the two concepts of shape "arm" and physical feature "beam."
2. Determine simplification level for the selected model (Figure 2-(b)).
   The designer determines appropriate simplification level for the selected aspect model, which is in this case a beam model. The designer selects physical phenomena that should be considered in the selected aspect model. In other words, some physical phenomena might be neglected. For instance, the designer can say that any bending is considered while vibration is not of interest in the arm design.
3. Exchange data among aspect models (Figure 2-(c)).
   The metamodel mechanism generates aspect models of the design object from the

metamodel. Suppose in the example of arm design, we might generate a bending aspect model and a geometric model. Since aspect models often require numerical information, the metamodel mechanism requests the designer to specify an appropriate aspect model to feed the required numerical information. For instance, since the generated bending aspect model needs dimensions of the beam, the designer specifies a solid modeler as a source to provide geometric information about the arm. Once this information is given, the metamodel mechanism can maintain consistently among the relationship between these two models.

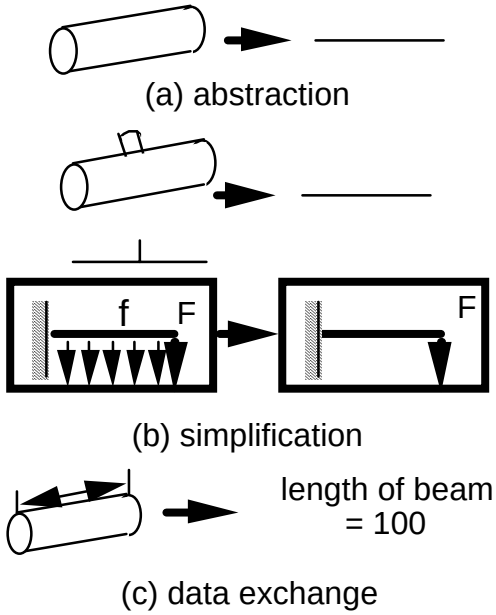


Figure 2 Modeling Process

3. The Pluggable Metamodel Mechanism

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

- Representation of knowledge and information in the metamodel.
  There can be different kinds of representation methods (data structure) for each attribute (e.g., solid, face, force, etc.). The system should absorb the differences in representation.
- Selection of an appropriate modeler.
  Since we do not directly describe the correspondence among modelers for exchanging data, the system should select an appropriate modeler and method to obtain data from the modeler.

3.1. Representation of data

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

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

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

3.2. Select an Appropriate Modeler

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

- Concepts used to build an aspect modeler
  This description is necessary to generate an qualitative aspect model from the metamodel.
- Data handled by an aspect modeler
  This description is used to identify an appropriate aspect model, when there is missing data.
- Data exchange method
  This description is used to feed data from a source attribute model to the aspect modeler.

Table 1 defines the knowledge about aspect modelers. We use usable concepts to generate a qualitative aspect model by abstracting the metamodel with these concepts. Available concepts are used to find out an appropriate aspect modeler for computing the value of attributes. We also define attribute translation methods from attribute(s) in the metamodel to an attribute in the modelers. These methods are represented by a conceptual network used to find out source attribute(s) from the metamodel by pattern matching of the graph. In addition, we define related concepts for filtering out unrelated concepts from the metamodel during construction of the initial qualitative aspect model. Table 2 is an example description about a beam modeler.

Note that these concepts are all defined in the concept dictionary which stores ontological knowledge about them. Because of this, when force is focused on, we can deal with any force regardless of its origin.

Table 1 Knowledge about Modelers

Name of the slot	Contents
Related concepts	List of concepts
Concepts used to build an aspect modeler	List of concepts
Data handled by an aspect modeler	List of concepts
Data exchange method	Attribute relationship graph and translating method

Table 2 Knowledge about Beam Modeler (Example)

Name of the slot	Contents
Related concepts	Entity, Relation, Force
Concepts used to build an aspect modeler	Beam, HingedSupport, ConcentratedForce, etc.
Data handled by an aspect modeler	ShearingForceDiagram, BendingMomentDiagram
Data exchange method	See Figure 3 for example

Figure 3 shows the definition of attribute translation method. Target attribute list (the upper part of center) represents the attribute list that is used in this beam modeler. Method list (beneath the target attribute list) represents candidate translation methods to get the selected target attribute (e.g., in this case Beam). Required concept graph (right upper corner) represents the required attribute value to exchange data. In this case to calculate the attribute value “Beam”, we need the attribute value of “Solid” which is an attribute of entity “Beam” to which the attribute “Beam” belongs. In the method browser, we define data translation method as a Smalltalk program. In this method, the standard methods described in Section 3.1. are used.

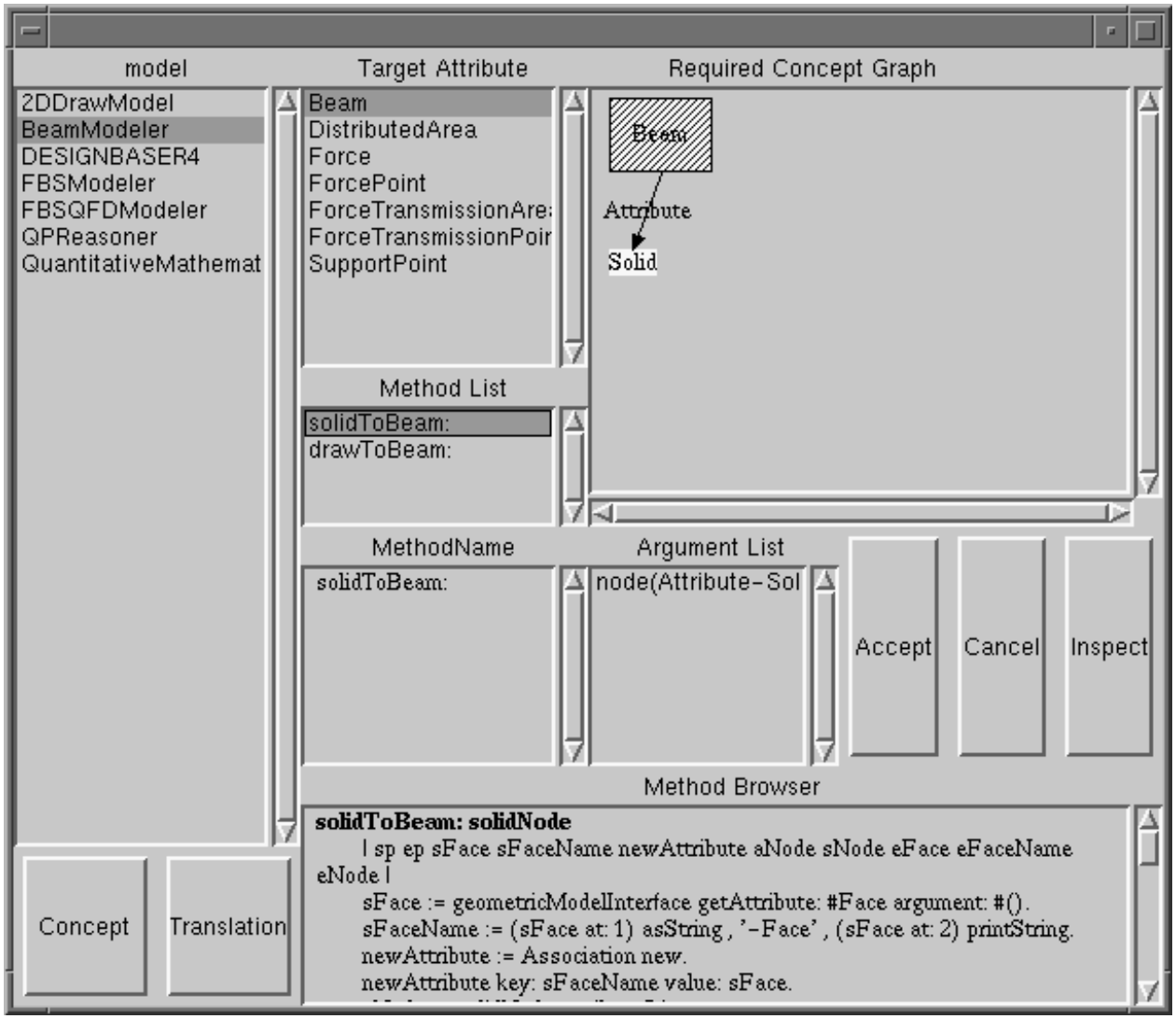


Figure 3 The Modeler Knowledge Browser

4. The Prototype System
4.1. The System Architecture

Based on the concept of the pluggable metamodel mechanism discussed in Sections 3, we have developed a prototype system that is capable of dealing with multiple existing design object modelers. The system is implemented in Visualworks[1] on a Sun workstation. Figure 4 depicts the architecture of the system. The current implementation of the prototype system is connected to five external design object modelers; i.e., a qualitative physics reasoning system [Kiriyama91], DESIGNBASE[2] (a solid modeler), a 2D draw modeler, the FBS(Function-Behavior-State) Modeler (Function modeling system) [Umeda96], and a beam modeler whose inference engine is Mathematica[3] (a symbolic processing mathematics system).

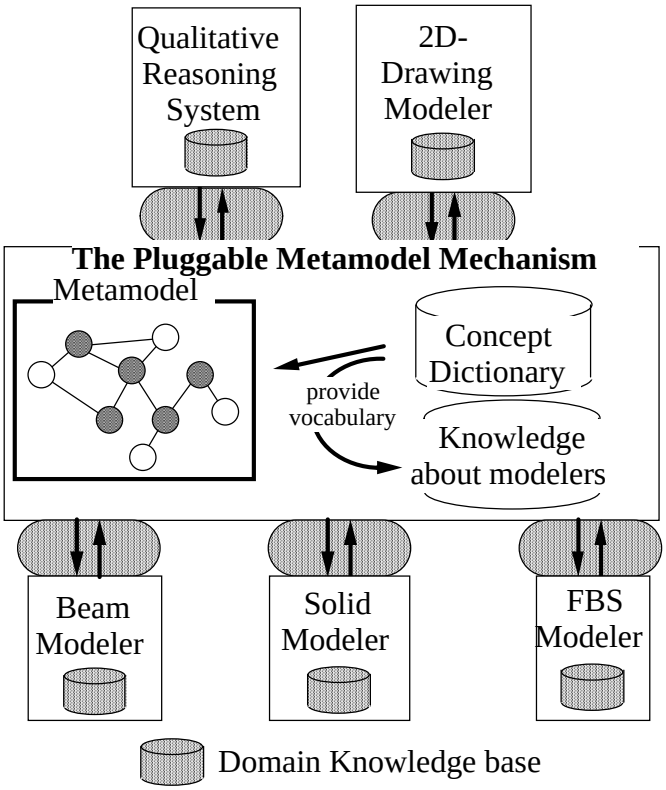


Figure 4 The Architecture of the Prototype System

[1] Objectworks is a registered trademark of ParcPlace Systems inc..
[2] DESIGNBASE is a trademark of Ricoh Company Ltd.
[3] Mathematica is a trademark of Wolfram Research Inc.

4.2. Example

Let us describe an example of designing a simple table driving mechanism. First, a designer uses the FBS modeler to decompose the functional specification and to build an initial qualitative model of the principal physical behavior and the structure of the design object. Figure 5 shows the created FBS model. The upper part represents the decomposed functional hierarchy, and the lower part represents the initial qualitative model of the physical behavior and the structure. From this initial model, the pluggable metamodel system reasons out possible physical phenomena that may occur to the mechanism. In this case, the system reasons out, for instance, gravity force on the ballscrew and on the table.

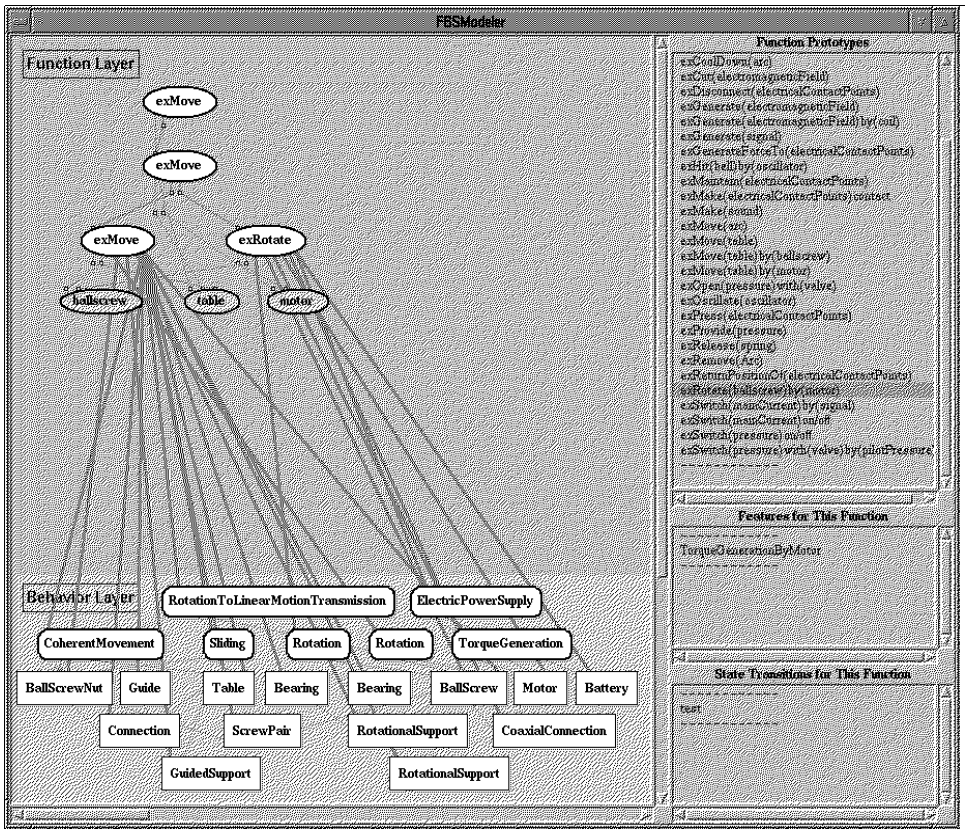


Figure 5 FBS Model of the Mechanism

This brings, the designer to notice that deformation might be important and he/she evaluates the deformation of the ballscrew with the beam model. Then the metamodel mechanism prepares abstract components for building a beam model; i.e., beam, distributed force, support, and so on (Figure 6). However, because at this stage there is no numerical information about the structure (length of beam, position of support, etc.), the metamodel mechanism cannot generate the beam model. Therefore, the system selects the appropriate modelers to handle these numerical information. In this case, the system selects the 2D draw modeler and the solid modeler with the knowledge about modelers. Then, the designer inputs shape data with one of these modelers (Figure 7). After inputting shape data, the system supplies the data to the beam modeler. Since it is difficult to determine the axis of beam automatically in this case, the system requests the

designer to specify the axis of the beam. Finally, new beam model is generated. Figure 8 depicts the shearing force diagram of the ballscrew computed by Mathematica.

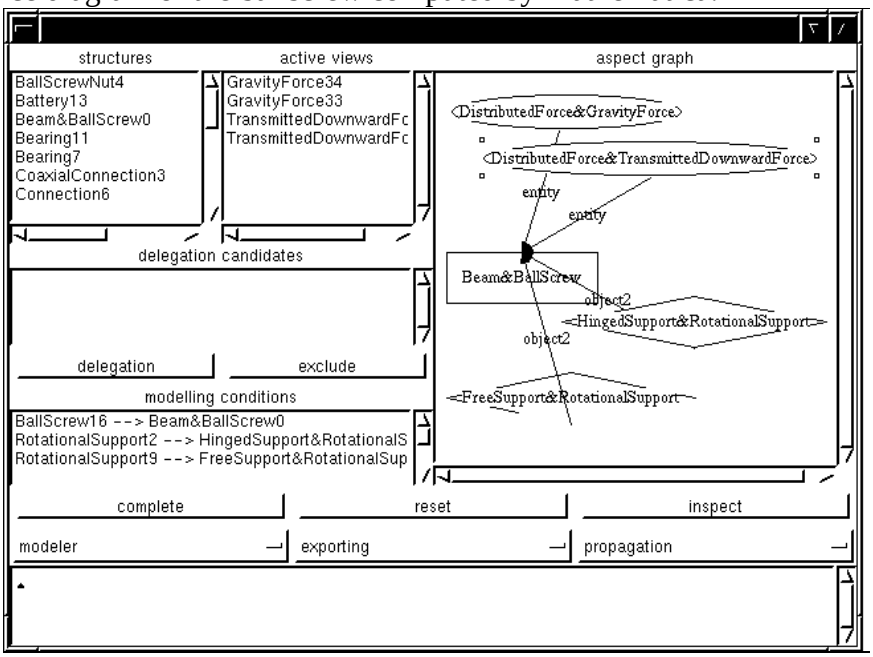
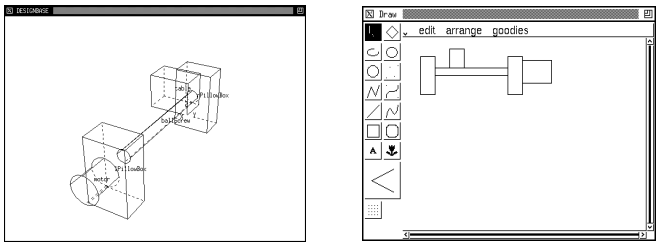
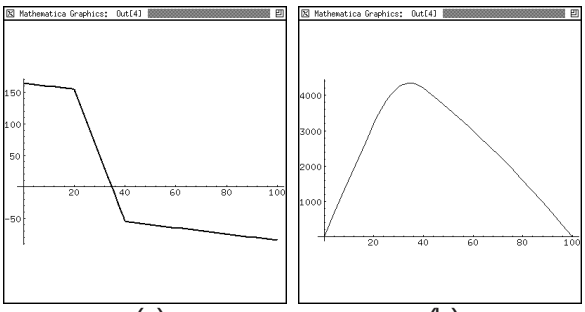


Figure 6 Qualitative Beam Model of the Ballscrew



(a) (b)

Figure 7 Geometric Model of the Mechanism
((a) solid model (b) 2D draw)



(a) (b)

Figure 8 Result of the Beam Model
(a) Sharing force diagram (b) Bending Moment Diagram

5. Related Work

STEP is an approach towards standard of product model. One problem we encountered is that STEP at present does not have ontological level description, based on which our system exchanges data among various modelers. In the future, we might be able to use as a standard of the attribute description in our framework.

PACT [Cutkosky93] is another framework for integrating existing external modeling systems. It is similar to our approach in that integrating is ontologically taken place based on the knowledge format; i.e., they use KIF (Knowledge Interchange Format) [Genesereth92] and Ontolingua [Gruber92]. We use concept dictionary as the most fundamental description. However they do not mention about the management of design object modeling process, although an intelligent agent oriented architecture is assumed.

6. Conclusions

This paper described a new integrated environment for design object modeling. This environment has an ability to plug in existing design object modelers and support modeling process through (1) symbolical operations to the model (such as abstraction and simplification) (2) exchanging data among models. We think this framework can contribute to build KIEF.

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

References

[Cutkosky93] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, and J. C. Weber: PACT: An Experiment in Integrating Concurrent Engineering Systems, *IEEE COMPUTER*, Vol. 26, No. 1, pp.28-37 (1993).

[Genesereth92] M. Genesereth and R. Fikes: Knowledge Interchange Format, Version 3.0 Reference Manual, Technical report logic-92-1, Department of Computer Science, Stanford University, Stanford (1992).

[Gruber92] T. R. Gruber: Ontolingua: A mechanism to support portable ontlogies, Technical report KSL91-66, Knowledge Systems Laboratory, Stanford University, Stanford (1992).

[ISO94] ISO10303-1: *Industrial Automation Systems and Integration - Product Data Representation and Exchange-, Part1: Overview and Fundamental Principles* (1994).

[Kiriyama91] T. Kiriyama, T. Tomiyama, and H. Yoshikawa: The use of qualitative physics for integrated design object modeling, In *Design Theory and Methodology (DTM '91)*, pp. 53-60, The American Society of Mechanical Engineers (ASME), New York (1991).

[Tomiyama89] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue: Metamodel: A Key to Intelligent CAD Systems, *Research in Engineering Design*, Vol. 1, No. 1, pp.19-34 (1989).

[Tomiyama94] T. Tomiyama, T. Kiriyama, and Y. Umeda: Towards Knowledge Intensive Engineering, In J. Sharpe and V. Oh (eds.): *Computer Aided Conceptual Design, Proceedings of the 1994 Lancaster International Workshop on Engineering Design CACD '94*, Lancaster University, Lancaster, UK, pp. 319-337 (1994).

[Tomiyama96] T. Tomiyama, Y. Umeda, M.Ishii, M. Yoshioka, and T. Kiriyama: Knowledge systematization for a knowledge intensive enginnerign framework, In T. Tomiyama, M. Mäntylä, and S. Finger(eds.): *Knowledge Intensive CAD-1, Preprints of the first IFIP WG 5.2 Workshop on Knowledge Intensive CAD 1,* pp. 33-52, Chapman & Hall (1996).

[Umeda96] Y. Umeda, M. Ishii, M. Yoshioka, and T. Tomiyama: Supporting conceptual design base on the Function-Behavior-State modeler, *Artificial Inteligence for Engineering Design and Manufacturing (AIEDAM)*, Vol.10, No. 4, pp. 275-288 (1996).

[Yoshioka93] M. Yoshioka, M. Nakamura, T. Tomiyama, and H. Yoshikawa: A design process model with multiple design object models, In *Design Theory and Methodology (DTM '93)*, pp. 7-14, The American Society of Mechanical Engineers (ASME), New York (1993).