Proceedings of DETC'01
ASME 2001 Design Engineering Technical Conference
and Computers and Information in Engineering Conference
Pittsburgh, PA, September 9-12, 2001

DETC2001/DAC-21155

PROPOSAL OF AN INTEGRATED DESIGN SUPPORT ENVIRONMENT BASED ON THE MODEL OF SYNTHESIS

Masaharu YOSHIOKA*
Research Center for Information Resources
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo 101-8430, Japan
Email: yoshioka@nii.ac.jp

Yutaka NOMAGUCHI
Tetsuo TOMIYAMA
Research into Artifacts, Center for Engineering
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo, 153-8904, Japan
Email: {noma,tomiyama}@race.u-tokyo.ac.jp

ABSTRACT

We propose a new integrated computational environment, called a Knowledge Intensive Engineering Framework (KIEF), to support design object modeling, during engineering design process that requires a variety of design object models, such as a geometric model, a control model, and a finite element model. KIEF is capable of guiding designers with design process knowledge based on a model of synthesis, of integrating multiple design object models, and of maintaining consistency among these models. The design process knowledge is a result of a research project we conduct to establish a model of synthesis. This model formalizes design process knowledge in a design process as operations to a multiple model-based reasoning system, such as KIEF, and from a language of synthesis. We describe a prototype system that incorporates design process knowledge to KIEF and illustrate an example design on it.

INTRODUCTION

In an engineering design process, designers use various design object models on many commercial design object modelers that are available to build, modify, operate, and evaluate such object models. Since these models are related to each other (e.g., shape data of a finite element model is closely related to shape data of a solid model), we need an integrated design support environment to manage these models.

A number of research efforts have been made to integrate design object models. In the earlier 1990s, several standard data representation format are proposed. IGES (Initial Graphics Exchange Specification) (ANSI/US PRO/IPO 100-1996, 1996) and STEP (Standard for the Exchange of Product model data) (ISO TC184/SC4, 1994) are typical examples of such standard formats. Nevertheless, these standard formats support only data exchange.

These ideas formed the concept of product model that is now commonly used for model management across a variety of applications in many commercial CAD systems. Furthermore in the 1990s, many commercial CAE (Computer Aided Engineering) tools were integrated as expansion modules in the geometry based CAD system, such as CATIA (IBM corporation, 2001) and Pro/Engineer (Parametric Technology Corporation, 2001). However, it is expensive and troublesome to plug in non-standard modules to these geometry-based integrated CAD systems, primarily due to the fact there does not exist a truly multipurpose, universal product model.

Artificial intelligence techniques also contributed to integrated design object modeling environments. For instance a high level modeling language such as Compositional Model Language (CML) (Falkenhainer *et al.*, 1994) was used to describe design objects that have many aspects, but this approach of symbolic representation of design objects was inappropriate to integrate existing modelers that are largely numerical data based.

We have already proposed a *Knowledge Intensive Engineer-*

*Address all correspondence to this author.

ing Framework (KIEF) that is a computational framework to integrate these design object models (Tomiyama *et al.*, 1996). KIEF employs a *Pluggable Metamodel Mechanism* (Yoshioka and Tomiyama, 1997) that represents and maintains relationships (e.g., causal dependency, translation, attribute, etc.) among concepts used in these models. This approach is different from product models in that it is based on knowledge level representation of design objects but with connections that allow exchanging of numerical data stored in various design object models.

However, when a design object should be evaluated, it still becomes difficult for designers to select an appropriate modeler from a variety of modeling systems offered by KIEF and to build a right model. Therefore, an integrated design support environment, such as KIEF, should be equipped with a capability of guiding designers during the modeling process including selecting an appropriate modeler and building a model. This guidance should be offered to the designers at the right time and in the right context considering the design process.

We have also conducted a research project to establish a model of synthesis (Yoshioka and Tomiyama, 1999). In this project, we formalized design processes with seven design knowledge operations based on a multiple model-based reasoning system, such as KIEF. Based on this model, we proposed model-based abduction (Yoshioka and Tomiyama, 2000) to formalize guidelines to integrate various model-based reasoning systems.

In this paper, we propose an integrated design support environment using KIEF as a design object representation system. We use design process knowledge proposed in the model of synthesis as knowledge for operations to the KIEF and formulate a language of synthesis. Finally, we describe a prototype system of this integrated design support environment with design process knowledge and illustrate an example design on it.

KNOWLEDGE INTENSIVE ENGINEERING FRAMEWORK (KIEF)

KIEF supports designers with multiple design object models by managing these models based on their dependency. There are two basic elements for the KIEF system.

- Knowledge base system
  It gives ontological definitions about concepts used in modeling systems.
- Pluggable metamodel mechanism
  This mechanism manages various models. It allows data sharing among modeling systems and maintains consistency among them.

Knowledge Base System

KIEF is requested to have a capability of handling a wide variety and a huge amount of design knowledge. To handle such a large amount of knowledge, traditionally engineering was organized as micro theories capable of generating and verifying a new solution. KIEF handles engineering knowledge as a set of micro theories that are codified in the modeling systems and integrates them by using common ontology and general knowledge about physical concepts.

Figure 1 depicts the component architecture of the knowledge base system with an example of gear transmission. The middle component, called a *concept base*, contains a basic ontology about physical concepts. In the concept base, physical concepts are organized as super (abstract) - sub (concrete) hierarchies and categorized into four types; i.e., entity, attribute, relation, and physical phenomenon.

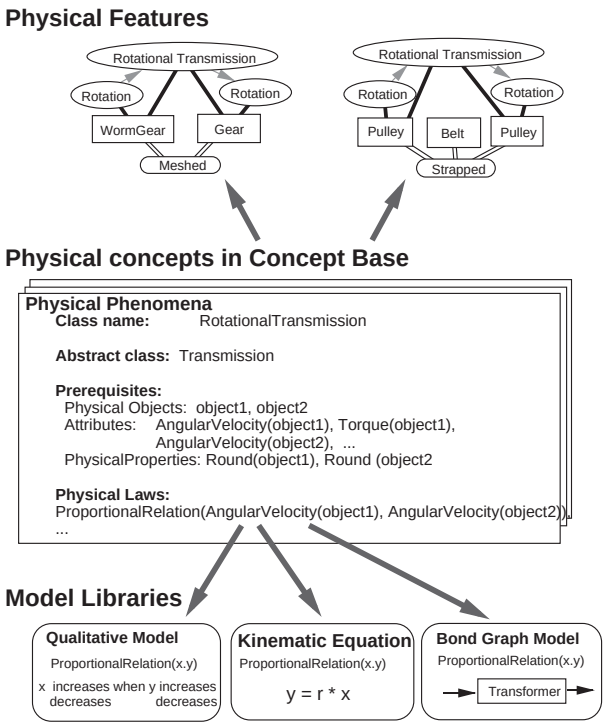

Figure 1. Knowledge Architecture for the KIEF

The upper side of Figure 1 contains a *physical feature*, such as a worm gear pair, which represents a combination of a set of entities and relations among the entities, and physical phenomena causally related to the entities. Physical features are used as building blocks for a physical model on this system. Physical concepts in the concept base provide a vocabulary to build the physical feature in this basic ontology layer. Model fragments for building a model with various model representations are stored in the model libraries with the relationship to physical phenomenon.

Pluggable Metamodel Mechanism

The pluggable metamodel mechanism is a computational system that integrates multiple design object models by using a metamodel (Figure 2) and knowledge about each modeler (Table 1). A metamodel represents relationships among physical concepts used in various design object models. Concepts are categorized into entity, relation, physical phenomenon and attribute. Relatinoships includes causality, entity-relation, entity-attribute, and so on. These definitions of physical concepts in this mechanism are given by the knowledge base system (Figure 3).

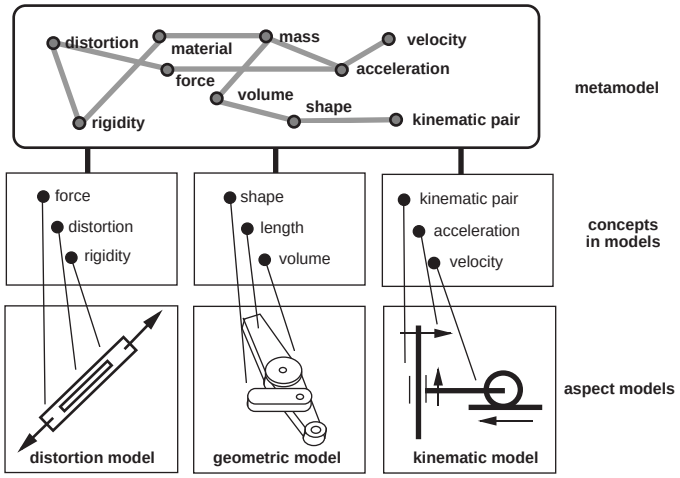

Figure 2. Metamodel (Tomiyama *et al.*, 1996)

Table 1. Knowledge about Modelers

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

The pluggable metamodel mechanism assists to build a model for each modeler using the information in the metamodel and in other modelers. A modeling process with the pluggable metamodel mechanism consists of two steps. The first step is a process to build a conceptual model for a particular modeler. In the second step, the pluggable metamodel mechanism obtains the information in other modelers to help the designer builds a model.

The pluggable metamodel mechanism stores information about the relationships among the concepts in the metamodel that

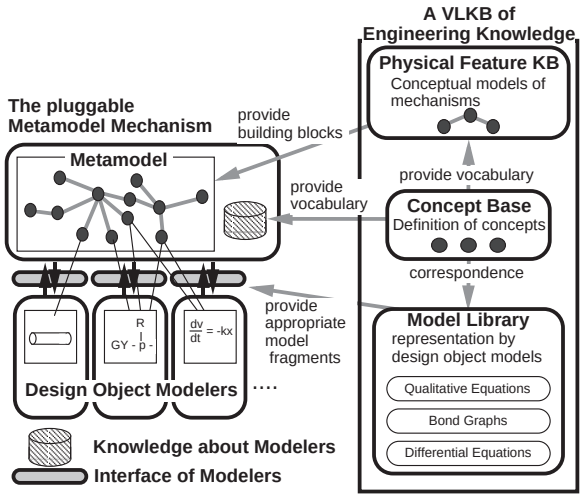

Figure 3. The Knowledge Intensive Engineering Framework (KIEF)

are used for the model. By using these relationships, the mechanism maintains the consistency among different models that collectively represent an identical design object.

THE MODEL OF SYNTHESIS

Engineering design consists of a variety of thought processes, but most of them can be categorized into either analysis or synthesis. Compared with analysis, synthesis is less understood and codified as a model. For instance, formal design methods proposed by German researchers frame engineering design as a process that begins with decomposition of given specifications into functional structure followed by embodiment using physical effects (VDI, 1977). This view of engineering design lacks notion of designer's thought process and understanding about synthesis, and does not help designers so much. It is, therefore, crucial to have a model of synthesis which is a core of design.

From 1996 to 2001, our group conducted a project entitled "Modeling of Synthesis" funded by JSPS. In this project, we first analyzed knowledge operations in design processes and proposed a reasoning framework of design (Yoshioka and Tomiyama, 1999). Next, we formalized abduction in design processes as model-based abduction. In the following section, we briefly review these formalizations.

Knowledge Operations in Design

From the observation of designers' activities, we identified following seven different knowledge operations and these operations do not come in a particular order.

- *Knowledge/Information Acquisition*

To acquire knowledge and information related to the problem.

- *Knowledge/Information Reorganization*
  To reorganize knowledge and information for a task.
- *Information Confirmation*
  To confirm information in one knowledge source by testing it against another information source.
- *Conflict Resolution*
  To resolve conflict among the different modeling systems.
- *Knowledge/Information Revision*
  To revise knowledge and information for integrating with other knowledge.
- *Solution Synthesis*
  To suggest a new solution for the problem.
- *Object Analysis*
  To analyze a design solution for evaluation.

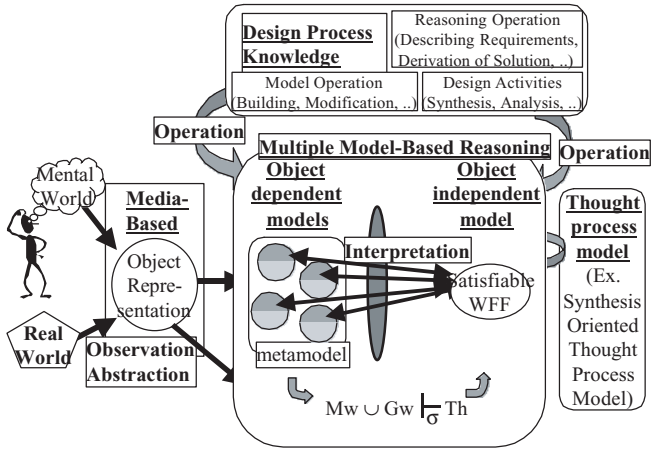

Figure 4. Reasoning Framework of Design (Yoshioka and Tomiyama, 1999)

The Reasoning Framework of Design

From the observation of designers' activities, we found out that the designer collects knowledge from various types of knowledge sources. In particular in detailed solution synthesis and object analysis processes, he/she collects and uses knowledge that is embedded in different engineering tools. These engineering tools act not only as working spaces but also as knowledge sources. Therefore, we can formalize design processes as multiple model-based reasoning processes.

Based on the discussion above, we propose a reasoning framework of design (Figure 4) to formalize seven knowledge operations proposed in the previous section. In this framework, object dependent models deal with a variety of knowledge and information about design objects, and they are integrated with the multiple model-based reasoning system. In addition, object independent models are controlled by thought process models.

In the knowledge/information acquisition process, a designer observes a physical world and collects knowledge and information about it. This process means the designer creates a mental model through which models in "media" are generated. These models in media include drawings and the design object itself. These models in media should be abstracted, captured, and codified as models on a computer. A different model can be generated for a different abstraction and codification system.

These codified models as a whole form a workspace for a designer by acquiring design information like design specification and his/her decision, and providing him/her with different perspectives about a design object. These codified models may correspond to object dependent tools and systems such as drawings, CAD systems, computational tools, databases, and knowledge bases. Therefore, this framework as a whole needs to be a multiple model-based reasoning system. Each of these models describes the design object from a particular perspective and stores background knowledge about that domain.

The designers' actual thought process will take place in the logical working space that is object independent. Any reasoning in this working space must be controlled by thought process models. (There can be different thought process models for analysis and synthesis, for instance.) The object dependent level models serve as "models" for the object independent model in the logic sense. In other words, the object dependent models determine truth-value in the logical working space of the object independent model.

By using the reasoning framework of design, we can codify the knowledge operations in design as operations to the object independent model and object dependent models.

Model-based Abduction

We believe abduction plays a critical role in design processes. We analyzed possible methods to formalize abduction and proposed *model-based abduction* that uses semantic information about design as operations to the multiple model-based reasoning system.

To handle semantic information about design, we analyzed design protocol and found out entities played a crucial role. First we found there was no knowledge directly coupling functional knowledge with attributive knowledge (Takeda *et al.*, 1990), (Tomiyama, 1995). Instead, the connection between functions and attributes is mediated by entities. Second, we found out that the most of entity concepts used in these protocols did not directly correspond to particular existing entities. These entities are abstraction of existing entities with selected important attributes, and design processes can be modeled as an evolution of this abstracted entity.

From these observations, we propose to organize design knowledge based on entity concepts. In addition, abduction in

design is a process to derive a design solution (entity) from functional requirements and/or required properties. Therefore, to formalize abduction in design, we classify the design knowledge (i.e., axioms) in the following two types where e denotes an entity, f a function, and p a property (attribute is a kind of property).

- Knowledge that describes an entity's functions: $e \rightarrow f$
- Knowledge that describe an entity's property: $e \rightarrow p$

Both of these types of knowledge cany be used bidirectionall. Suppose an entity that should fulfill a functional requirement f_i. This process should derive an entity e_i using the first type knowledge and abduction. Then, the second knowledge is used to obtain property information about the solution.

This formalization can classify the nature of various knowledge based systems for design. For example, let us consider a catalog retrieval system that can retrieve design solutions (entities) from functional specifications. Even though the system is based on a simple database lookup method, the system does perform synthesis by means of non-logical abduction. This classification implies we can integrate the results of logical abduction and non-logical abduction to derive final design solutions and that reduce the computational complexity of abduction.

SYNTHESIS LANGUAGE TO DESCRIBE DESIGN PROCESS KNOWLEDGE

Based on the discussion in the previous section, we propose a *synthesis language* for describing design process knowledge. The synthesis language controls multiple model-based reasoning system in the reasoning framework of design according to the status of the design object information.

Operations in the Model of Synthesis

As we discussed in the model-based abduction section, we use the model-based reasoning systems (object dependent models) as knowledge sources and the object independent model at logical level manages the information in each system. So, we discuss the correspondence between seven knowledge operations and operations in the model-based abduction algorithm proposed in (Yoshioka and Tomiyama, 2000). We also discuss how to realize the algorithm as operations to the multiple model-based reasoning system.

1. Set up the initial requirements in the object independent model.

   Knowledge operation *Knowledge/Information Acquisition*
   Operation in the reasoning system Represent initial requirements in an object dependent model and export the information to the object independent model.

2. Select a model-based reasoning system that can derive abstract entity concepts from the knowledge about the systems. If there is no more knowledge base to be used, then end.

   Knowledge operation *Knowledge/Information Confirmation, and Knowledge/Information Reorganization*
   Operation in the reasoning system First, select unsolved design problems. Second, select model-based reasoning systems that can deal with the problems and select one system from them.

3. Build a model as a solution candidate with the selected system.

   Knowledge operation *Knowledge/Information Confirmation, Knowledge/Information Reorganization, and Solution Synthesis*
   Operation in the reasoning system Build a model on the selected system by using information in the object independent model and use a model to conduct abduction type operations.

4. Analyze the model using knowledge stored in the system. This is basically deduction and the results (i.e., the model's properties) are added to the object independent model.

   Knowledge operation *Knowledge/Information Confirmation, Knowledge/Information Reorganization, and Solution Analysis*
   Operation in the reasoning system Use a model to conduct deduction type operations and export new obtained information to the object independent model.

5. Confirm the validity of object independent model. This means comparison of the derived properties with the requirements. If it is OK, the current result is asserted and go to step 2. If it fails, resolve conflict.

   Knowledge operation *Knowledge/Information Confirmation, Conflict Resolution, and Knowledge/Information Revision*
   Operation in the reasoning system Check consistency in the object independent model. If a conflict is found in the model, the system checks the source of the conflict by using logical dependency information. Then select a model-based reasoning system that can revise the source of the conflict and revise it.

In order to realize the operations to the multiple model-based system in this algorithm, we define primitive operations to the object dependent models and object independent model. In addition to that, to navigate the designer in the model-based abduction algorithm, we also need to describe knowledge about what to do next according to the status of the design object information. Furthermore, this language should have capability of interacting with the designer, such as to input the design specification and to select one solution from solution candidates.

Based on these, we discuss the guidelines to select primitive modeling and logical operations for each type of knowledge

operations as follows.

- Operations for object dependent models
  These include the following four types of operations. However, in order to support operations for object dependent models, it is necessary to have detail knowledge for each modeler.

  1. Selection of a model-based reasoning system
  2. Building a model on the selected system
  3. Using model on the selected system
  4. Operations to exchange the information between the object dependent model and object independent modele

- Operations for object independent models
  Most of the operations for the object independent model manage information in the object dependent models and support the designer to select an appropriate model-based reasoning system for the problem.
  To do so, the design object information should be categorized into three; i.e., function, entity, and attribute. In addition, we define a category of requirements to check the status of the design object information. By using this category and knowledge about modelers (Table.1), we can support the selection of appropriate modelers for the problem. For example, let us consider that the designer wants to use an abduction system to derive an entity from the functional requirements. Modelers that can deal with the functional requirements and derive an entity from them are the appropriate one.
  Since design is a process to derive an entity from the design specifications given as functions and attributes, logical dependencies among these concepts are necessary to check whether all of the design specifications are satisfied or not.
  We also use abstract entity concepts to formalize the model-based abduction. Since those abstract entities do not have enough information to manufacture, it is necessary to check the entity descriptions from the view point of manufacturing. The iteration of the abduction type reasoning and deduction type reasoning is controlled by a design history management system that records changes of the design object information.
- Operations for user interaction
  There are two types of interaction. One is the operation to select one from multiple candidates. The other is the operation for the object dependent models. Since we do not formalize detail operations in these models, the system should have a capability of stopping the navigation while the designer uses each model-based reasoning system and of resuming the navigation after the usage. For the interaction between the real world, we use a particular model-based reasoning system to input new acquired information.

Descriptions of Design Process Knowledge in the Synthesis Language

Based on the guidelines we proposed in the previous section, we propose primitive operations (see Appendix A) to describe the design process knowledge.

Since each step of the model-based abduction algorithm corresponds to a set of knowledge operations in the model of synthesis and these operations are related to each other, it is necessary to describe the relationships between different operations. For example, step 3 (Building a model) requires selection of a modeling system in step 2.

We have decided to use a Lisp-like format to describe the design process knowledge description. There are two components in one knowledge. One is a condition element that checks the status of the design object. The other is an operation list that operates design object models. Figure 5 shows an example of the design process knowledge for solution synthesis.

We describe design process knowledge for the following six operations as well.

1. Input requirement
2. Conflict Resolution
3. Make prototype and experiment
4. Solution analysis
5. Solution synthesis
6. Information revision

In order to navigate the designers' thought process, we set a priority for each of the operations. In the list above, a lower number means higher priority. For example, since "Input specification" has a high priority, the system asks the designer to input specification. After inputting the specification information, the condition for the "Input specification" is not satisfied because the specification infomraion is already given by the designer. At this moment, "Solution synthesis" is a possible operation with a high priority. After conducting "Solution synthesis," the conditions for "Solution analysis" are satisfied and analyze the solution candidates.

AN INTEGRATED DESIGN SUPPORT ENVIRONMENT

System Architecture

Based on the previous discussion, we propose an integrated design support environment (Figure 6). There are two different levels of inference in this environment. One is the object level inference in which the system operates the design object information. The other is action level inference in which the system suggests possible operations to the design object models and a user selects one operation from them.

For object level inference, we use KIEF as a multiple model-based reasoning system in the model of synthesis. Since the metamodel in KIEF can be represented as a logical formula, we use the metamodel as an object independent logical model. Other

Conditions

```
(or (not (isEmpty: (reject:by: (getCurren-
tRequirement) isJustified:)))
 ; Not all of requirements are satisfied
 ; by the design object information (not (isEmpty:
(reject:by: (getCurrentDesignObjectDescrip-
tion) isManufacturable:))))
 ; Not all of the entity description has enough
 ; information to manufacture
```

Operations

```
((setq CandidateProblems (reject:by:
(getCurrentRequirement) isJustified:))
 ; Select requirements that are not satisfied
 ; and set them as candidate problems. (setq RelatedMod-
elers (relatedModelers:type: CandidateProb-
lems synthesis))
 ; Select modelers that can be used for solving
 ; the candidate problems and set them as candidates.
(setq SelectedModeler (selectOne-
From:message: RelatedModelers 'select one
modeler for synthesis.'))
 ; Select one modeler from the candidates
(setq NewModel (makeModelerOn: SelectedMod-
eler))
 ; Make model on the selected modeler
(interactive (useModeler: NewModel))
 ; Use model on the modeler
 ; System stops and resumes when the designer finish
 ; to use the modeler
(setq UsedInformation (getUsedInformation:
NewModel))
 ; Get information that is used for making a model
(setq NewInformation (getModelerInformation:
NewModel))
 ; Get information that is represented in the modeler
(modifyInformation: (difference:from:
UsedInformation NewInformation) (differ-
ence:from: NewInformation UsedInformation))
 ; Propagate the difference to the object independent model
```

Figure 5. Example Description of Design Process Knowledge

modeling systems such as solid modeler are integrated as object independent models. We also implement a design process control system.

For action level inference, this system checks the status of KIEF to evaluate the design process knowledge described in the proposed synthesis language.

A design process with this environment proceeds as follows.

1. Checking the status of KIEF
   The design process control system checks the status of KIEF and selects possible candidate design operations.
2. Selection of a design operation

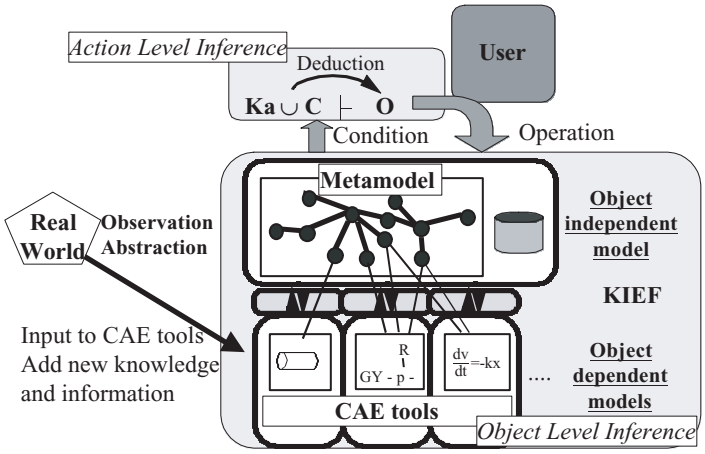

Figure 6. System Architecture of Integrated Design Support Environment

The user selects one of the candidate design operations and the design process control system controls the operation in the KIEF system.

3. Interactive operation
   When the design process control system requires an interactive operation to KIEF, the system interrupts the operation and asks the designer for an input. When a series of operations is over, the system checks the status again and goes back to step 1.

Example

We have implemented a CAD system for the laser stereolithography design based on this environment. Since one of the main features of the system is the design process navigation based on the design process knowledge, we used a real design history (Xie *et al.*, 1999) and compared the system's behavior with this design history.

The main purpose of this design process is to improve surface accuracy of the laser stereo lithography system and the designer generates four types of solution candidates.

To implement the CAD system, we had to add the following modelers.

- Function-Behavior-State (FBS) modeler (Umeda *et al.*, 1996) for functional design
- Qualitative Physics Abduction System (QPAS) (Ishii and Tomiyama, 1996) to derive solution candidates from the first principles
- Solid modeler (Pro/Engineer)
- Catalog retrieving system
- Surface tension analyzer

Figure 7 shows the hardcopy of the user interface. The upper window shows the status of the design object information and

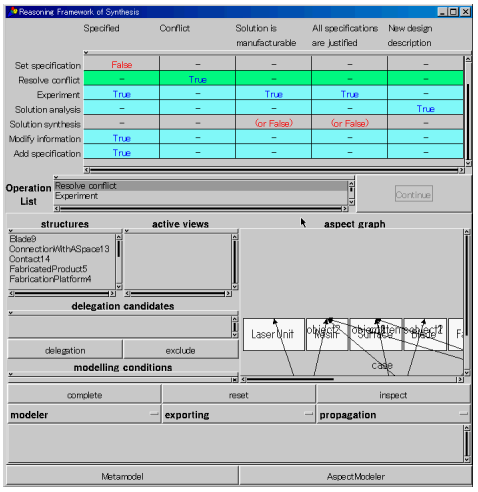

Figure 7. System Hardcopy

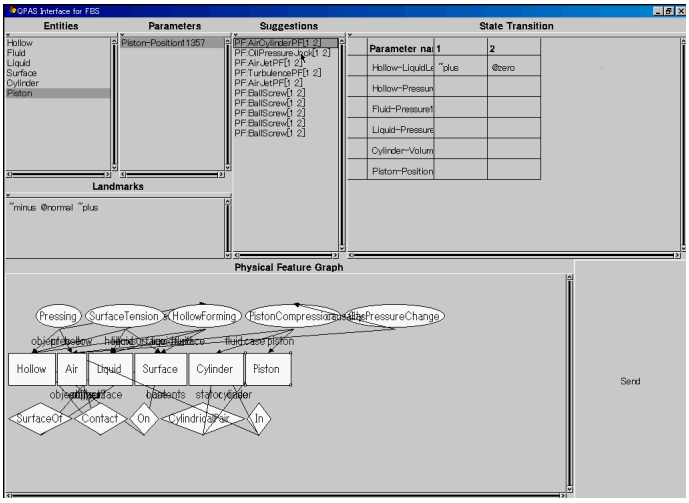

Figure 9. Result of the QPAS

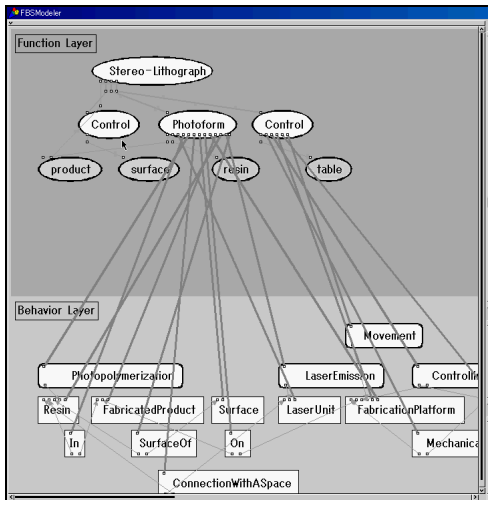

Figure 8. Result of the FBS modeler

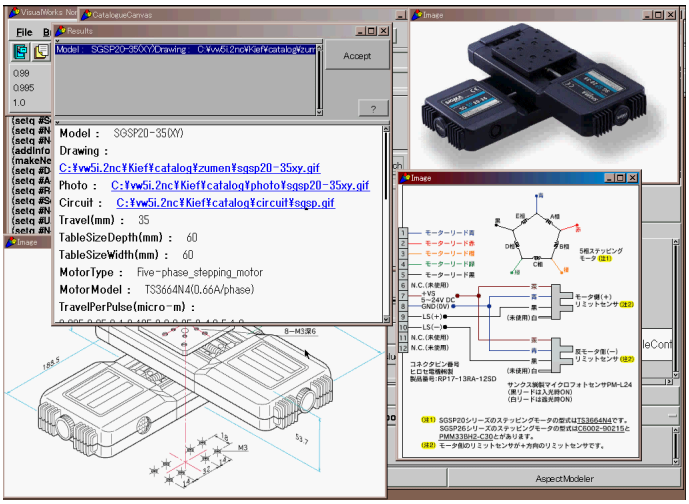

Figure 10. Result of the catalog modeler

the middle window shows the possible operations. The bottom window is used to represent design object information.

The system integrates different types of abduction systems to generate different solution candidates. In the most abstract level, design problems are given in the functional description and the designer uses the FBS modeler to arrive at alternative design solutions (Figure 8).

When the design problems are given as state transitions (changes of the attribute values according to time), the system can generate solution candidates with QPAS (Figure 9). Since this system generates combinations of different mechanisms as solution candidates, much of them are meaningless.

In detail level design, when the design problems are given as selection of a detailed entity from the abstract entity concepts, the designer can use the catalog retrieval system (Figure 10) to select an appropriate mechanism.

The system can also support numerical analysis based synthesis. Figure 11 shows the result of the surface tension analysis. Based on the analysis, the designer modifies the radius of the nozzle and that may cause the shape information modified in the solid modeler (Figure 12).

Discussion

From the example above, we can find out that this system has capability to integrate different types of abduction systems to solve the design problem.

We also compared the design process conducted by the sys-

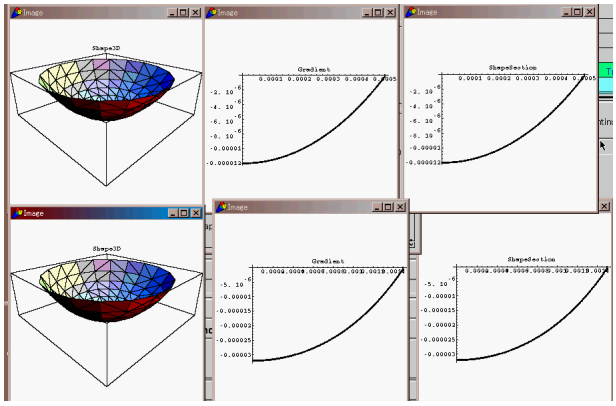

Figure 11. Result of the Surface Tension Analysis

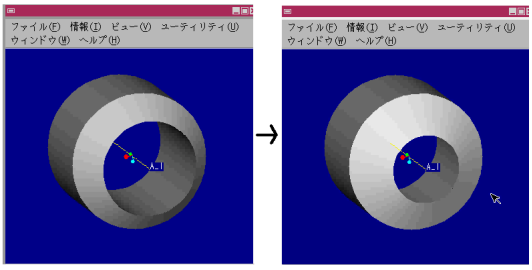

Figure 12. Modification of the Shape Information based on the Surface Tension Analysis

tem with a real design process and found out that the system can trace most of the design process. However, since we did not implement enough knowledge to conduct the whole design process, some details were left untouched. The detail of the comparison are discussed in (Tsumaya *et al.*, 2001).

RELATED WORK

The Distributed Object-based Modeling and Evaluation (DOME) framework integrates distributed design object models (Pahng *et al.*, 1998). This framework has flexibility to integrate design tools. However, since this framework does not have ontological knowledge, the designer should set identification of the same data in the different design object models manually.

Another approach is generating integrated design object models with a high level modeling language such as Compositional Model Language (CML) (Falkenhainer *et al.*, 1994), and Adaptive Modeling Language (AML) (TechnoSoft Inc.). For example, Ozawa *et al.*, (1999) proposed to use CML for implementing a design system for optical pick-up heads of DVD players. Bhungalia *et al.*, (2000) used AML for an integrated design system of hypersonic vehicles. The motivation and vision presented in this paper have similar themes. However, since they only focused on the modeling part, they did not deal with design process knowledge to control modeling processes.

CONCLUSION

In this paper, we first analyzed the research results of the Moodeling of Synthesis project and proposed a synthesis language that can describe design process knowledge in a multiple model-based reasoning system. Then, we proposed an integrated design support environment based on KIEF. In this framework, the synthesis language controls the operations to KIEF. We also implemented the design system for laser stereo-lithography on this environment and found various types of abduction can be integrated in this environment.

For the future work, we plan to apply this environment to other design problems for further verification.

ACKNOWLEDGMENT

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

REFERENCES

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

Bhungalia, A. A, Zweber, J. V, and Stevenson, M. D. *Integrated aerodynamic and geometric modeling for hypersonic vehicle design.* In Proceedings of the 2000 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, New York, 2000. The American Society of Mechanical Engineers (ASME). DETC2000/DAC-14267 (CD-ROM).

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

IBM corporation. *CATIA solutions homepage.* http://www-3.ibm.com/solutions/engineering/escatia.nsf/Public/catia_overview, 2001

Ishii, M and Tomiyama, T. *A synthetic reasoning method based on a physical phenomenon knowledge base.* In Sharpe, J, editor, AI System Support for Conceptual Design, Proceedings of the 1995 Lancaster International Workshop on Engineering Design, pp. 109–123, 1996.

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

Ozawa, M, Cutkosky, M. R, and Howley, B. J. *Model sharing among agents in a concurrent product development team.* In

Finger, S, Tomiyama, T, and Mäntylä, M, editors, Knowledge Intensive Computer Aided Design, pp. 143–165. Kluwer Academic Publishers, Dordrecht, 1999.

Pahng, F, Senin, N, and Wallace, D. *Distributed object-based modeling and evaluation of design problems.* Computer-aided Design, Vol. 30, No. 6, pp. 411–423, 1998.

Parametric Technology Corporation. *Pro/Engineer solution.* http://www.ptc.com/products/proe/index.htm, 2001.

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

TechnoSoft Inc.. AML web page. http://www.technosoft.com/products.html.

Xie, T, Murakami, T, and Nakajima, N. *Micro photoforming fabrication using a liquid hollow shaped by pressure difference and surface tension.* International Journal of the Japan Society for Precision Engineering, Vol. 33, No. 3, pp. 253–258, 1999.

Tomiyama, T. *A design process model that unifies general design theory and empirical findings.* In 1995 Design Engineering Technical Conferences Vol. 2, pp. 329–340, New York, 1995. The American Society of Mechanical Engineers (ASME).

Tomiyama, T, Umeda, Y, Ishii, M, Yoshioka, M, and Kiriyama, T. *Knowledge systematization for a knowledge intensive engineering framework.* In Tomiyama, T, Mäntylä, M, and Finger, S, editors, Knowledge Intensive CAD-1, IFIP TC5/WG 5.2 Workshop Series on Knowledge Intensive CAD, pp. 33–52. Chapman & Hall, 1996.

Tsumaya, A, Nomaguchi, Y, Yoshioka, M, Takeda, H, Murakami, T, and Tomiyama, T. *Verification of a model of synthesis -the methods forverification and results.* Submitted to International Conference on Engineering Design, ICED 2001.

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

VDI-Gesellschaft. *VDI-2222: Konstrution und Entwicklung.* VDI-Verlag, Düsseldorf, 1977

Yoshioka, M and Tomiyama, T. *Pluggable metamodel mechanism: A framework of an integrated design object modelling environment.* In Bradshaw, A and Counsell, J, editors, Computer Aided Conceptual Design '97, Proceedings of the 1997 Lancaster International Workshop on Engineering Design CACD'97, pp. 57–70. Lancaster University, 1997.

Yoshioka, M and Tomiyama, T. *Toward a reasoning framework of design as synthesis.* In Proceedings of the 1999 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, New York, 1999. The American Society of Mechanical Engineers (ASME). DETC99/DTM-8743 (CD-ROM).

Yoshioka, M and Tomiyama, T. *Model-based abduction for synthesis.* In Proceeding of the 2000 ASME Design Engineering Technical Conference & Computers and Information in Engineering Conference, New York, 2000. The American Society of Mechanical Engineers (ASME). DETC2000/DTM-14553 (CD-ROM).

Appendix A: Primitive Operations in Synthesis Language

- Set operation
  - Check the status (empty set and subset of other set) of set(s)
  - Set calculation (intersection, difference, and addition)
  - Select elements that meet given condition
- Object dependent models
  - Selection of a model-based reasoning system
  - Build a model
  - Select information that is used for model building
  - Use a model
  - Propagate new added information in the model
- Object independent model
  - Add information from the object dependent models
  - Select categorized information (function, attribute, entity, and requirement) from the design object information
  - Check the consistency
  - Check whether requirements are justified with other information or not
  - Check whether the entity description has enough information to manufacture or not
- Interactive operation
  - Select one element from the set.
  - Stop the navigation and resume it.
  - Make prototype
  - Conduct experiment
- History operation
  - Make new world with new information
  - Change previous world