4

Knowledge systematization for a knowledge intensive engineering framework

T. Tomiyama, Y. Umeda, M. Ishii, M. Yoshioka
Department of Precision Machinery Engineering
The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan
Telephone: +81-3-3812-2111, Fax: +81-3-3812-8849
{tomiyama, umeda, ishii, yoshioka}@zzz.pe.u-tokyo.ac.jp

T. Kiriyama
Research into Artifacts, Center for Engineering (RACE)
The University of Tokyo
Komaba 4-6-1, Meguro-ku, Tokyo 153, Japan
Telephone: +81-3-5453-5891, Fax: +81-3-3467-0648
kiriyama@race.u-tokyo.ac.jp

Abstract
This paper proposes knowledge intensive engineering that is a new way of engineering activities in various product life cycle stages flexibly conducted with more knowledge to create more added value. Knowledge representation and modeling issues are discussed and a cooperative multiple intelligent agent architecture based on multiple ontology is proposed for building a Knowledge Intensive Engineering Framework (KIEF). KIEF can be used as a knowledge intensive CAD for knowledge intensive design of knowledge intensive machines. This demonstrates the power and usefulness of knowledge intensive engineering. It is also discussed that to achieve knowledge intensive engineering, systematization of knowledge is an essential process to allow intelligent agents to share accumulated knowledge.

Keywords
Engineering knowledge, knowledge systematization, knowledge intensive engineering, computer-aided design (CAD)

1 INTRODUCTION

Knowledge intensive engineering is a new style of engineering based on intensive use of various kinds of engineering knowledge aiming at creating more added value (Tomiyama 1994). Concurrent engineering addresses the very same issue of flexibly using various kinds of knowledge and aims at improving manufacturing processes by taking, for example, production issues into design consideration. Knowledge intensive engineering further elaborates this idea, such that having a framework for knowledge intensive engineering allows designers and engineers to create more added value in various stages of product life cycle stages including as designing, manufacturing, operations, and maintenance.

At the University of Tokyo, our group has been studying a wide variety of topics related to engineering design. Among other things, developing an intelligent CAD system was a primary goal (Veth 1987; Xue *et al.* 1992). This aims at integrating and making flexible use of various kinds of design knowledge and intelligently assisting designers, by giving advises and suggestions and by checking errors based on design process knowledge. The Knowledge Intensive Engineering Framework (KIEF) is a computational framework for knowledge intensive engineering and is a natural extension of the concept of the intelligent CAD system (Tomiyama 1994; Tomiyama *et al.* 1994; Yoshikawa *et al.* 1994).

Note that knowledge intensive engineering is not just a new variation of knowledge engineering, but a new style of engineering based on the recognition that knowledge is the source of added value. Since added value can be generated only from knowledge, obviously we need more knowledge in various aspects of engineering. KIEF should allow generation and flexible exchange of design knowledge and information that will be further used in various stages of the life cycle. It is not just another 'knowledge-based design' system that assists designers in an intelligent manner.

The rest of the paper is organized in the following manner. Chapter 2 discusses the concepts of knowledge intensive engineering and examples of how knowledge intensive engineering can increase added value of various engineering activities. Knowledge intensive engineering requests that various kinds of engineering knowledge must be systematically organized and installed in a computational framework such as KIEF. To do so, we need to systematize engineering knowledge. Chapter 3 examines this issue of knowledge systematization, while Chapter 4 deals with the concept of knowledge intensiveness that is the key issue of knowledge intensive engineering. Chapter 5 discusses the architecture of KIEF based on the ideas of cooperative multiple intelligent agent architecture and its kernel knowledge representation and modeling methodologies. Chapter 6 concludes the paper.

2 THE CONCEPTS OF KNOWLEDGE INTENSIVE ENGINEERING

Engineering design requires a wide variety of knowledge about product life cycle aspects. Therefore, an advanced information support infrastructure for engineering design should deal with not only design knowledge, i.e., about requirements, functions, physical phenomena, and devices, but also knowledge about manufacturing processes, operations, diagnosis and maintenance, and even disposal and recycling.

This coincides with the recent trends that product life cycle issues are paid more attention due to, first, environmental problems and, second, high-tech products that should be installed, operated, maintained, and recycled exclusively by manufactures. Concurrent engineering nowadays addresses not only manufacturability issues but also maintainability, serviceability, recycleability, etc. (Tomiyama 1995). This indicates that manufacturing industry is now extending its scope from just design and manufacture to the entire life cycle stages of products including marketing, material acquisition, design, production, logistics, installation, operation, maintenance, reclamation, recycling, and discarding. The famous Japanese 'design-in' activity is a supporting evidence of the fact that such intensively collected knowledge is more effective than a distributed, independent set of knowledge.

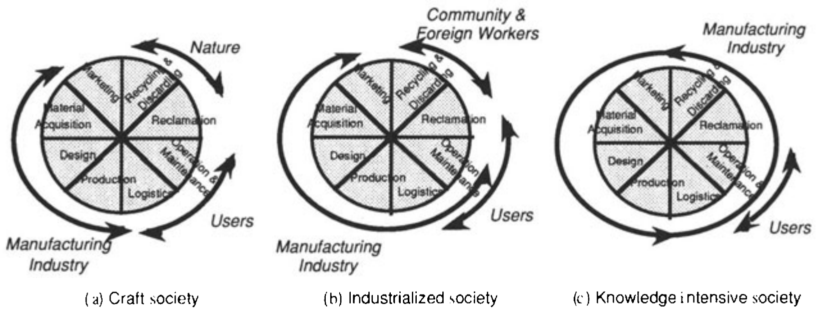


Figure 1 Craft society, industrialized society, and knowledge intensive society.

Thus, it is necessary to use and develop advanced computer tools for facilitating mutual communications among those involved in every stage of the product life cycle. Figure 1 depicts how manufacturing industry extended its scope over product life cycle in its history and will extend in the near future (Tomiyama and Baba 1993).

This idea resulted in the concept of **knowledge intensive engineering** which is a new way of engineering activities in various product life cycle stages conducted with more knowledge in a flexible manner to create more added value. It is crucial to share and reuse various kinds of knowledge involved in the product life cycle. Technically speaking, in case of, e.g., **knowledge intensive design**, this boils down to integration and management of various kinds of models to synthesize an artifact, to analyze its properties, and to evaluate its performance against requirements under certain circumstances. These models are not simply connected to each other but integrated, so that mutual data exchange and model sharing are possible.

Knowledge intensive engineering has another goal to increase added value of engineering activities. Recently, we see various kinds of advanced mechatronics machines emerge and intelligence is becoming an indispensable part of innovative products and manufacturing processes. One of the bottlenecks in developing such 'smart machines' is software development. If KIEF can provide a software engineer with design information including functional and behavioral requirements, device control information, and physical constraints regarding sensors and actuators, the software development can be greatly assisted or even automated (Shimomura *et al.* 1993). We might call this process **knowledge intensive software development.**

Operation tasks require different models from design, such as a control model, sensory data models, etc., so do maintenance tasks. For instance, **knowledge intensive maintenance** is a process in which various kinds of models for maintenance are used to diagnose a target machine more accurately and correctly than traditional association based diagnosis expert systems and to generate more useful information such as inspection instructions and repair plans. Information obtained through operation and maintenance should be further fed back to reliability design. In this way, KIEF can serve also as a basis for knowledge intensive operation and maintenance. Figure 2 illustrates how knowledge intensive engineering is useful in a product life cycle.

This can be further extended to the **knowledge intensive enterprise** concept which is based on a smooth flow of design knowledge from the design department to the production, sales, and other departments and even to products themselves as embedded intelligence. For instance, we have developed self-maintenance photocopiers that are typical examples of such innovative, **knowledge intensive machines** of which intelligence comes directly from design knowledge (Umeda *et al.* 1992; Umeda *et al.* 1994b).

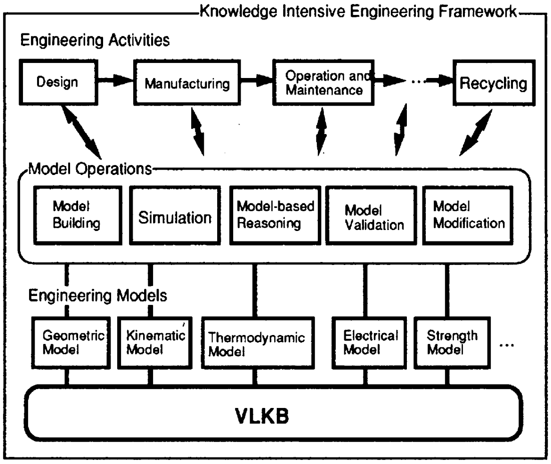


Figure 2 Product life cycle and knowledge intensive engineering.

3 KNOWLEDGE SYSTEMATIZATION

Knowledge intensive engineering requests that various kinds of engineering knowledge must be systematically organized and installed in KIEF. To do so, we have been conducting various projects to collect design knowledge (Kiriyama *et al.* 1992; Tomiyama *et al.* 1994). The most significant result of these knowledge collection projects is that, if knowledge is **systematized**, the task boils down to a matter of actual collection. Otherwise, it is extremely difficult even to collect knowledge (Tomiyama *et al.* 1992).

3.1 Classification of knowledge

We consider that knowledge systematization consists of the following processes, *viz.*, setting up a view, articulation, codification, crystallization, verification, and reusing and sharing of knowledge. Before we examine these in detail, we illustrate our general view about knowledge.

Generally speaking, knowledge has two forms, i.e., recognized knowledge and unrecognized knowledge. Another dimension of knowledge is tacit knowledge and codified knowledge. Tacit knowledge is a form of knowledge that is explicitly or implicitly recognized by human and used for reasoning but very difficult to describe. Codified knowledge is a form of knowledge that is described in any human-recognizable form. This means that there are three kinds of knowledge as combinations of these two dimensions (Figure 3); recognized and codified, recognized and tacit, and unrecognized and tacit. Unrecognized and codified knowledge is meaningless.

The primary goal of knowledge systematization is to convert recognized and tacit knowledge to recognized and codified knowledge. However, there can be various degrees of codification. For instance, information stored in a database is computable. Textbooks can be machine-readable, but might not be computable. Since our ultimate goal is the development of KIEF, computable knowledge is the best form of knowledge. For obtaining computable knowledge, we need further treatment.

Science in general is an effort to create recognized and codified knowledge from tacit knowledge. While science sometimes focuses on even unrecognized knowledge, systematization of knowledge does not. On the other hand, science does not necessarily aim at creating computable knowledge.

In this paper, we discuss conversions from recognized and tacit knowledge to recognized and codified, and preferably, further to computable knowledge as a working definition of systematization. Figure 4 depicts the knowledge systematization process.

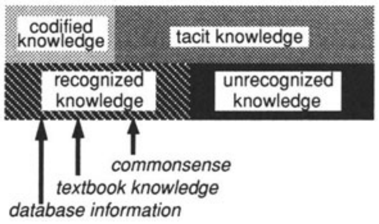


Figure 3 Classification of knowledge.

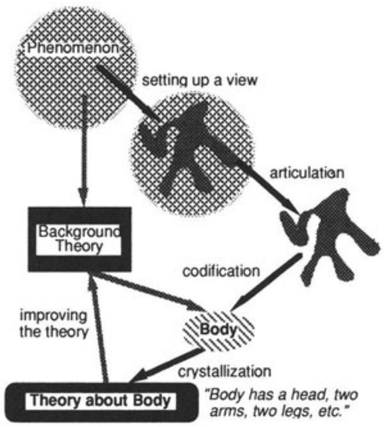


Figure 4 Knowledge systematization process.

3.2 Setting up a view

In order to convert recognized and tacit knowledge to codified, first we need to set up a view. This can be best explained by how modeling works (Tomiyama *et al.* 1990). In science and engineering, a model is created by observing a phenomenon based on a theory (Figure 5). A kinematic model is generated based on kinematics, using only the vocabulary and concepts of kinematics. In this regard, modeling is a kind of filtering through observation, such that a generated model becomes complete as far as the theory is concerned.

There can be many different models for a single phenomenon. This implies that, when converting tacit knowledge into codified knowledge, we need to have a pre-described set of concepts. This set is called a view and often associated with a well-defined theory that describes relationships among those concepts. Systematization of knowledge, thus, begins with defining such a view.

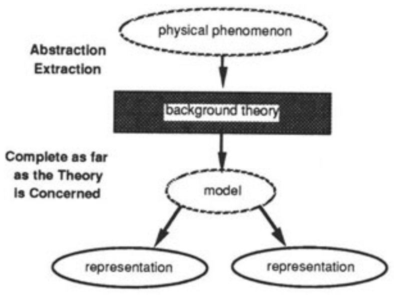


Figure 5 Modeling.

3.3 Articulation

Once we set a view together with a background theory associated with the view, we try to identify instances of concepts that belong to the view in the observed phenomenon. This process is called articulation in which unorganized instances of concepts are given representations chosen from the vocabulary. Knowledge after articulation is thus recognized knowledge, or knowledge before articulation is tacit knowledge.

3.4 Codification

After articulation, we have a set of instances that were given representations. Since in the process of setting up a view we also have a background theory that defines relationships among these instances, this set of instances can now be organized and given structure. In artificial intelligence, structure is often represented by entities and relationships among entities, and attributes of entities and relationships. At this stage, the phenomenon is codified and our knowledge about the world can increase through knowledge collection. However, codified knowledge is only a collection of facts observed in the phenomenon. A fact is a (partial) account of the phenomenon within the theory. Rules obtained in a knowledge acquisition process are examples of such codified knowledge.

3.5 Crystallization

Codification process generates only pieces of factual knowledge. This type of knowledge is extremely useful when massively stored in a database system. However, such knowledge can result in hard-fail. A knowledge based system can hard-fail, when it encounters a situation that is not described in the knowledge base. In other words, any knowledge base has its limitations and lacks robustness. An approach to removing this limitation is either to broaden the range that the knowledge base covers or to generalize the knowledge. Projects to construct a large scale knowledge base, such as Cyc (Lenat 1995; Lenat and Guha 1989), are examples of the former approach. The latter is achieved by two ways: one is to use deeper knowledge and the other is to use a general, abstract knowledge. Crystallization is a process to generate general and abstract knowledge from purely factual knowledge in an organized manner. Such general, abstract knowledge is called a theory. This theory is different from the background theory in that it is backed up by observed facts.

An important lesson that we learnt from our project to build a large scale knowledge base is that crystallization is needed to organize the codified knowledge (Tomiyama *et al.* 1994). This implies that a brute force approach to collect knowledge eventually fails. Knowledge collection must be done in a systematic and organized way.

3.6 Verification of knowledge

The next step is to verify the collected knowledge with any known theories. Self-verification is a test of the codified knowledge or the crystallized theory against the background theory, while other kind of verification is a test against any other known theories. If there found any contradiction, we need to modify, improve, or adjust the background theory or the articulation scope, or to redo the codification or crystallization processes.

3.7 Reusing and sharing of knowledge

Knowledge intensive engineering requires reusing and sharing of knowledge that are the eventual goal of knowledge systematization. In order to reuse and share knowledge, we need not only knowledge representation techniques but also a common knowledge description format such as KIF (Genesereth and Fikes 1990). Systematization of knowledge cannot be accomplished without such a method. However, we do not discuss this issue here.

4 KNOWLEDGE INTENSIVENESS

4.1 Large scale knowledge bases

Systematization of knowledge is achieved by conversions from recognized and tacit knowledge to recognized and codified knowledge. There can be various types of knowledge according to the degree of systematization, which include for example:

- factual database;
- large scale knowledge base;
- domain dependent, object level theory;
- computable knowledge.

It is often advocated that large scale knowledge bases are useful for engineering applications including design, manufacturing, operations, and maintenance, because these activities require an extremely huge amount of and various kinds of knowledge. Similarly, a considerable body of current AI research centers on the approach of very large-scale knowledge bases (VLKB), such as Cyc. These projects aim at building powerful AI systems, first by collecting a large number of knowledge chunks from ontological, fundamental common knowledge to domain dependent specific knowledge, and second by providing mechanisms for reusing and sharing knowledge (Neches *et al.* 1991). The KIF Project advocates the idea of **knowledge standard** for exchanging knowledge. Obviously, VLKB should go hand in hand with knowledge standards.

4.2 Knowledge intensiveness

Not only quantity of knowledge but also quality of knowledge in terms of sharability and reusability of knowledge is crucial, because knowledge intensive engineering aims at both amount and flexibility of knowledge. Knowledge must be well organized and can be flexibly applied to different kinds of applications. Figure 6 schematically depicts measures of the knowledge intensiveness concept. Certainly, knowledge intensiveness refers to the amount in a chunk of knowledge. Abstractness is an index of generality or universality of a chunk of knowledge. Width here shows the domain that is covered by a chunk of knowledge. Intensity means how knowledge is well systematized within a chunk.

We view that knowledge intensiveness comes not only from the amount of knowledge but also from the way in which knowledge is distributed among cooperative intelligent agents. Flexibility in Figure 6 means ease of using a chunk in combination with others.

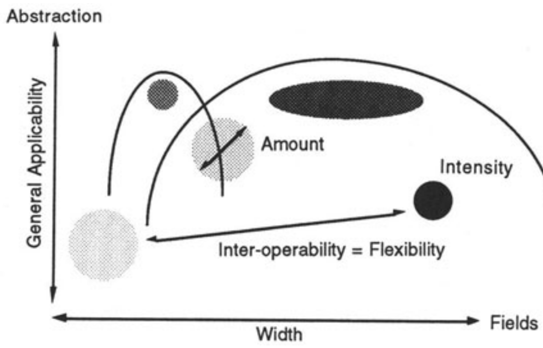


Figure 6 Knowledge intensiveness.

4.3 Knowledge sharing and reuse

Figure 7 compares three different types of knowledge sharing architecture. Figure 7 (a) depicts a situation with independent (and probably irrelevant) knowledge bases as a result of collecting independent 'micro theories.' In this case, the strength of knowledge is just a sum of each of independent knowledge bases. Integrated knowledge bases can be represented in Figure 7 (b) in which knowledge bases can be applied to various situations and the strength of knowledge is maximum. However, this requires to have a platform with a uniform language. The Cyc project is an example of this approach. In Figure 7 (c), independent knowledge bases can communicate and form an inter-operable situation, although the strength of knowledge might be weaker than that in Figure 7 (b). The entire knowledge base is a federation or a set of loosely coupled intelligent agents. For instance, the PACT project employs this strategy (Cutkosky *et al.* 1993). As a platform, they use KIF and Ontolingua (Gruber 1993).

The second architecture can suffer from the lack of uniform knowledge representation. Unless carefully designed, the platform language cannot cover everything. The alternative third approach may have less problem in this regard, if the framework is enough abstract to incorporate different types of ontology each agent talks about. However, it does not mean that we can completely avoid the problem, because communication among agents requires at least to understand what other agents are talking about.

From our previous experiences with building a large scale knowledge base, we have found out that reasoning based on single ontology is less flexible and that describing everything in a single ontology is extremely difficult. This single ontology situation corresponds to Figure 7 (b). Instead, we need to have a system that allows reasoning based on multiple ontology in a cooperative multiple intelligent agent architecture depicted in Figure 7 (c).

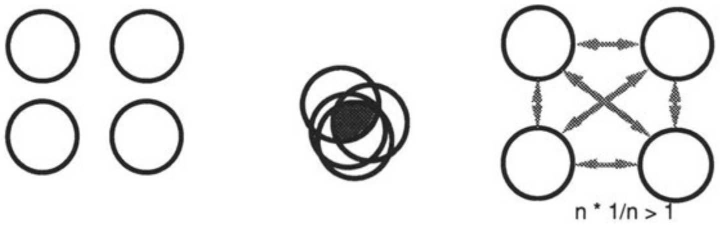


Figure 7 Knowledge sharing architectures.

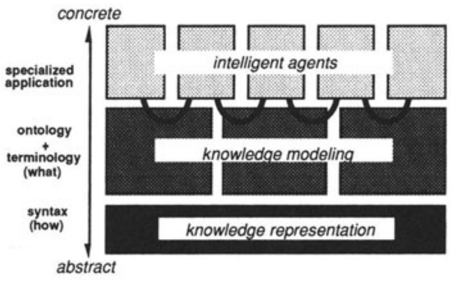


Figure 8 Knowledge representation, modeling, and agents.

4.4 Cooperative multiple intelligent agents

Figure 8 depicts how we can build a cooperative architecture composed of communicating intelligent agents based on multiple ontology. Intelligent agents are computer systems or human agents of any kind including knowledge based systems (in most ordinary sense), database systems, simulation systems, and specific purpose computational tools. The simplest one could be a program that performs arithmetic.

At the bottom of the architecture, i.e., the syntax level, there is a system for knowledge representation, which is less critical than the knowledge modeling level that describes 'ontology.' While the knowledge representation level is more concerned with how, the knowledge modeling level represents what to be modeled and defines the basis of communication for intelligent agents. Knowledge modeling refers to such techniques as model based reasoning, qualitative physics, etc. We call this level ontology. The ontology at this intermediate level might include (but not necessarily be limited to) causality, spatial and temporal relationships, and actions in many cases.

The ontology level describes concepts with fixed sets of concepts. These sets define 'terminology.' Terminology is a network of conceptual relationships (or a dictionary), and at this moment, there are six categories of terminology, i.e., entity, relation, attribute, physical property, physical phenomenon, and function.

Anything undefined in this level should be resolved by agents that provide problem solving capabilities. Each agent, therefore, has its own ontology (or problem solving mechanism) inside it and does not have to understand detailed knowledge and algorithms with which other specialized agents perform their tasks. This is an advantage of this approach when compared with, e.g., the integrated knowledge representation approach in Figure 7 (b).

The intermediate layer should also contain information about what other agents know about. This includes information about the input-output relationships of an agent and information about concepts that an agent knows, although some details can and should be omitted.

5 A COMPUTATIONAL FRAMEWORK FOR KNOWLEDGE INTENSIVE ENGINEERING

Based on the ideas discussed in the previous chapters, our group at the University of Tokyo is currently developing KIEF. Its overall architecture is depicted in Figure 9. The framework is developed on Objectworks\Smalltalk (from ParcPlace Systems, Inc.).

5.1 Metamodel mechanism

At the core of the system, we have the metamodel mechanism with VLKB of engineering knowledge (Ishii *et al.* 1995; Kiriyama *et al.* 1991; Kiriyama *et al.* 1992; Tomiyama *et al.* 1990; Tomiyama *et al.* 1994; Tomiyama *et al.* 1992; Yoshikawa *et al.* 1994). Database management is done by an object oriented database system (VERSANT from Versant Technology, Inc.). Intelligent agents are models of an object to be modeled for various engineering activities.

The metamodel mechanism is based on common ontology and terminology for model management to allow intelligent agents to communicate each other. It is called metamodel, because it models various kinds of models. The metamodel mechanism integrates various kinds of models through the concept network dynamically created by envisioning. As a result of envisioning, suppose that we know a physical phenomenon may qualitatively happen. To evaluate this, we need to generate a dedicated qualitative model concerned about that phenomenon and may further need to generate a quantitative model. The metamodel mechanism feeds information necessary to do so and automatically generates required models (Kiriyama *et al.* 1991). These generated models can share and exchange knowledge and information through the concept network. This is a substantially different approach from, e.g., product data exchange technology such as STEP (ISO 1990).

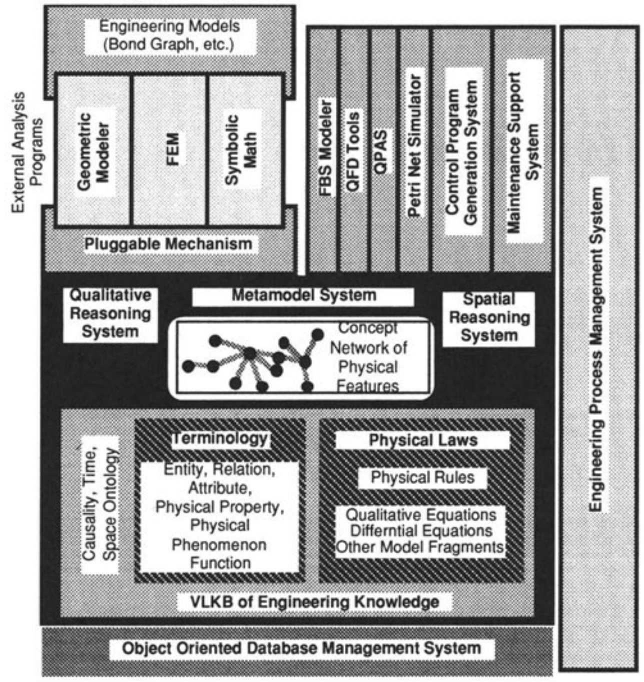

Engineering Models (Bond Graph, etc.)
External Analysis Programs
Geometric Modeler
FEM
Symbolic Math
Pluggable Mechanism
FBS Modeler
QFD Tools
QPAS
Petri Net Simulator
Control Program Generation System
Maintenance Support System
Engineering Process Management System
Qualitative Reasoning System
Metamodel System
Concept Network of Physical Features
Spatial Reasoning System
Causality, Time, Space Ontology
Terminology
Entity, Relation, Attribute, Physical Property, Physical Phenomenon Function
Physical Laws
Physical Rules
Qualitative Equations Differntial Equations Other Model Fragments
VLKB of Engineering Knowledge
Object Oriented Database Management System

Figure 9 Architecture of the knowledge intensive engineering framework.

5.2 Kernel knowledge representation

The knowledge modeling of the metamodel mechanism is based on multiple ontology but a single terminology. Currently, there are four different types of ontology either implemented or planned. The first type is causality based on Qualitative Process Theory (QPT) (Forbus 1984). The second type is temporal ontology in addition to temporal notions used by a qualitative reasoning system based on QPT. The third type of ontology is spatial. These two types of ontology, temporal and spatial, allow reasoning with dedicated reasoning systems. We plan to incorporate the fourth type of ontology that is about action. We call these four types of ontology CATS (Causality, Action, Time, and Space) ontology.

In Figure 9 showing the architecture of KIEF, VLKB stores fundamental engineering knowledge based on the ontology described above. There are fundamental concepts that are commonly used within the system. These are called terminology and include concepts about entities, relations, attributes, physical properties, physical phenomena, and function. Each of these concepts is separately categorized and forms a static conceptual network.

Entity

An entity represents an atomic, physical object in our representation and corresponds to an 'individual' in QPT. Entities are organized in an abstract-concrete hierarchy. The hierarchy allows multiple inheritance, so that an entity can be categorized into more than one class; for example, a spur gear can be a subclass of the mechanical rotational part and of the metallic object.

Relation

A relation represents a relationship among entities, such as 'on,' 'above,' 'below,' 'support,' and 'connection.' In other words, it defines structure among entities. Relations work exactly in the same way as predicates in predicate logic. Relations can be hierarchically defined.

Relations are one method to introduce the spatial and temporal ontology into the system. For instance, a relation 'on' between two entities, A and B, does not have any meaning, unless there is a system that can verify that A is actually on B. This is a sort of symbol grounding problem and can be solved only by introducing an external reasoning system that can confirm if this relation holds.

Attribute and physical property

An attribute is a concept attached to entities and takes a value to indicate the state of entities, such as 'position,' 'temperature,' and 'mass.' A physical property is a concept that describes generic characteristic of entities, such as 'elastic' and 'magnetized.' A physical property is associated with a set of attributes that indicate degree of the physical property. For example, Young's modulus indicates elasticity. The knowledge base also has knowledge about differential relations between attributes, and 'has' relations between entities and physical properties.

Physical phenomenon

A physical phenomenon designates physical laws or rules that govern behaviors of the object and corresponds to a view of QPT. Figure 10 shows a screen hardcopy of the physical phenomenon browser defining a physical phenomenon in which a spring is reacting to external force. A physical phenomenon is defined by the following slots:

- Name of the physical phenomenon.
- Super (or abstract) physical phenomena.
- Prerequisites for the physical phenomenon to become activated in terms of entities together with their attribute definitions, relations, and physical properties. Entities are specified as pointers to link instances of entities for delegation when building physical features.
- Physical rules that define relationships among the parameters and define influences that can happen when the physical phenomenon is activated.

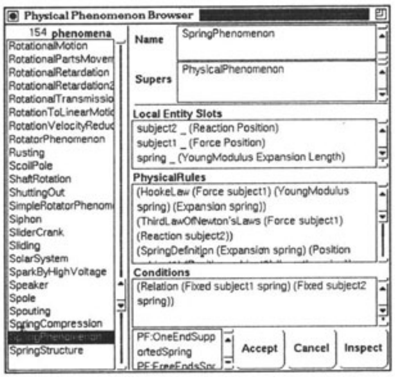
Physical laws

The physical law knowledge base contains two types of knowledge. One is the physical rule knowledge base that stores the unique names of physical laws and the attribute concepts used in the laws. The names of physical laws keep track of the same physical law represented in different design object models. The other is the model libraries that store typical model fragments of a design object model that relate physical concepts (such as entities, relations, physical phenomena, etc.) and design object models. For example, an equation

$$f = m\alpha$$

is a fragment of a dynamics equation model that describes Newton's second law. These fragments are used as building blocks to generate design object models. The parameters used in a fragment corresponds to attribute concepts, so that, after a design object model is generated, the metamodel mechanism can maintain relationships among the attribute concepts in the metamodel and the parameters in design object models for sharing information.

Figure 11 is a screen hardcopy of the physical rule browser that defines Newton's second law of motion.



Physical Phenomenon Browser

154 phenomena

RotationalMotion
RotationalPartsMovem
RotationalRetardation
RotationalRetardation2
RotationalTransmissio
RotationToLinearMotic
RotationVelocityReduc
RotatorPhenomenon
Rusting
ScollPole
ShaftRotation
ShuttingOut
SimpleRotatorPhenom
Siphon
SliderCrank
Sliding
SolarSystem
SparkByHighVoltage
Speaker
Spole
Spouting
SpringCompression
SpringPhenomenon
SpringStructure

Name: SpringPhenomenon

Supers: PhysicalPhenomenon

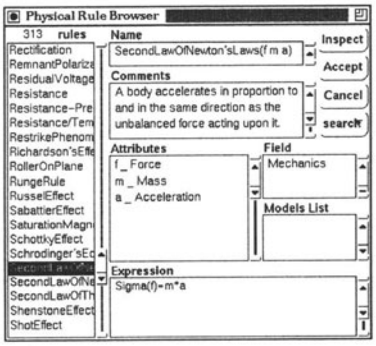
Local Entity Slots
subject2 _ (Reaction Position)
subject1 _ (Force Position)
spring _ (YoungModulus Expansion Length)

PhysicalRules
(HookeLaw (Force subject1) (YoungModulus spring) (Expansion spring))
(ThirdLawOfNewton'sLaws (Force subject1) (Reaction subject2))
(SpringDefinition (Expansion spring) (Position

Conditions
(Relation (Fixed subject1 spring) (Fixed subject2 spring))

PF:OneEndSupportedSpring
PF:FreeEndsSpr

Figure 10 The physical phenomenon browser.



Physical Rule Browser

313 rules

Rectification
RemnantPolariza
ResidualVoltage
Resistance
Resistance-Pre
Resistance/Tem
RestrikePhenom
Richardson'sEffe
RollerOnPlane
RungeRule
RusselEffect
SabattierEffect
SaturationMagn
SchottkyEffect
Schrodinger'sEq
SecondLawOfNe
SecondLawOfNe
SecondLawOfTh
ShenstoneEffect
ShotEffect

Name: SecondLawOfNewton'sLaws(f m a)

Comments: A body accelerates in proportion to and in the same direction as the unbalanced force acting upon it.

Attributes
f _ Force
m _ Mass
a _ Acceleration

Field: Mechanics

Models List

Expression: Sigma(f)=m*a

Figure 11 Physical rule browser.

Physical features

A physical feature describes a situation in which something happens. Figure 12 shows representation of a physical feature that includes the physical phenomenon of gear transmission. A physical feature is represented by a network of physical phenomena, entities, and relations. Links between a physical phenomenon and entities represent that the particular physical phenomenon occurs to a set of particular entities. Links among physical phenomena represent causal dependencies, which means that activation of one of the physical phenomena causes activation of others. In this sense, a physical feature is a sort of scenario of the situation; a set of causally related physical phenomena happening simultaneously or sequentially states, e.g., a situation that 'fuel burns, generates heat, and warms up air.' Links between a relation and entities represent that the entities have the relation.

Figure 13 illustrates how the knowledge of gear transmission can be represented in KIEF. We can use 'rotational transmission' instead of 'gear transmission.' The functional relations of gear's velocity are described in various representations in the model libraries. The qualitative reasoning system based on QPT takes physical features as an input and reasons about all feasible behaviors of the object. Its output is a concept network of relevant entities, relations, attributes, etc., and a sequence of state transitions that represents a set of feasible behaviors. This concept network is used to identify relevant physical phenomena that might occur to the design object and that are useful information for, e.g., building models and maintaining consistency among them.

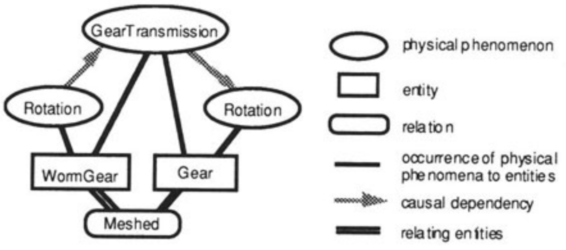
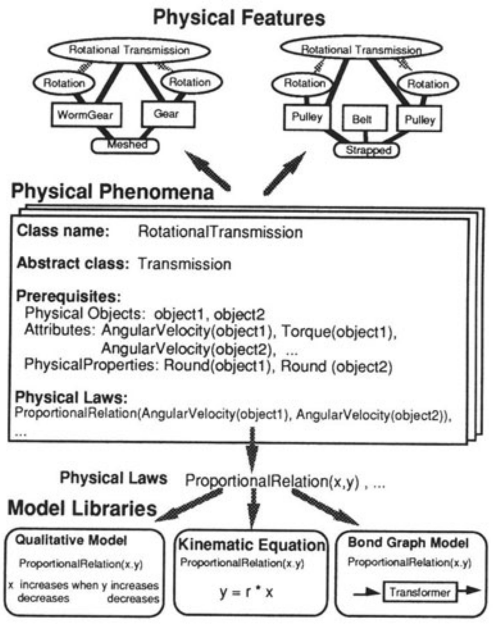


Figure 12 A physical feature representing gear transmission.



Class name: RotationalTransmission

Abstract class: Transmission

Prerequisites:
Physical Objects: object1, object2
Attributes: AngularVelocity(object1), Torque(object1), AngularVelocity(object2), ...
PhysicalProperties: Round(object1), Round (object2)

Physical Laws:
ProportionalRelation(AngularVelocity(object1), AngularVelocity(object2)), ...

Physical Laws ProportionalRelation(x,y) , ...

Qualitative Model: ProportionalRelation(x,y) — x increases when y increases, decreases decreases

Kinematic Equation: ProportionalRelation(x,y) — $y = r * x$

Bond Graph Model: ProportionalRelation(x,y)

Figure 13 Description of gear transmission.

5.3 Function

In the context of knowledge intensive engineering activities, functional knowledge plays a fundamental role. For example, in engineering design, function is the requirement to be satisfied by the design solution. We employ the FBS (Function-Behavior-State) modeling (Umeda *et al.* 1990) as the basis for function modeling.

We define a function as a subject description of a behavior, while a behavior is defined as a set of state transitions. A function is represented in the form of 'to do something.' Behaviors and state transitions are reasoned out by the qualitative reasoning system as described in the previous section. A function prototype defines how this function can be hierarchically decomposed into sub functions or which behaviors (as a sequence of state transitions) can realize this function (called F-B relationship). The function hierarchy is composed of either abstract-concrete relations or whole-part relations. An example of decomposition into sub functions is that a super function 'to move table' can be decomposed into three sub functions, namely, 'to stop table,' 'to move table by motor,' and 'to stop table.' Figure 14 shows a complete FBS model of the contactor (an electric relay).

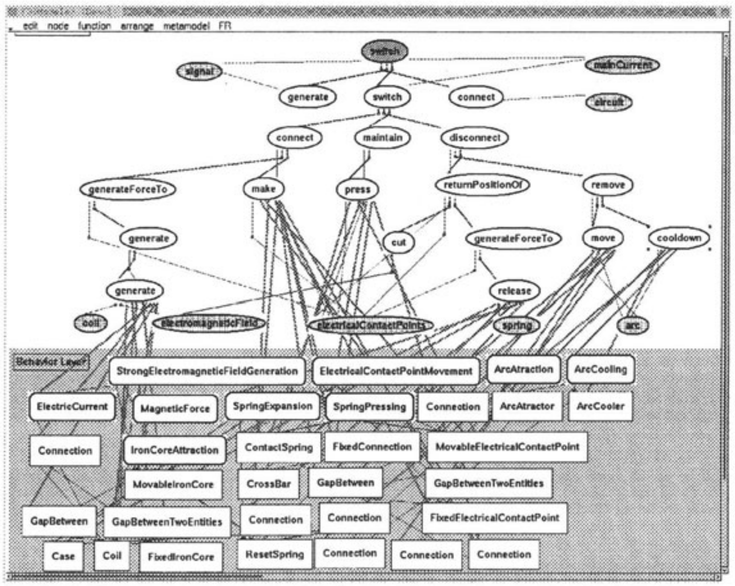


Figure 14 The FBS modeler.

5.4 Pluggable metamodel mechanism

Models used for different engineering activities can have different kinds of problem solving mechanisms and different scales of scope, aspect, granularity, and abstraction. For instance, a geometric modeling system has precise geometric information based on algebraic geometry, while some other subsystem may require only topological (or connectivity) information. To absorb differences in the abstraction levels, the framework has a pluggable metamodel mechanism through which existing external modelers can be plugged into KIEF (Yoshioka *et al.* 1993) (Figure 9). Since different types of information cannot be integrated easily, we need information describing conceptual relationships between different types of information.

So far, we have incorporated the following agents and modelers into the framework in addition to the qualitative physics based model which is dealt with by the qualitative reasoning system.

- The FBS modeler for functional modeling. This system can be used for functional design including innovative design for high reliable machines based on the idea of function redundancy (Tomiyama *et al.* 1993; Umeda *et al.* 1992; Umeda *et al.*. 1994).
- A geometric modeling system (DESIGNBASE from Ricoh, Co.).
- An FEM system (MODIFY developed by Ohtsubo *et al.* (1993)).
- A 2-D spatial reasoning system to symbolically deal with the spatial ontology.
- A tool for Quality Function Deployment (QFD) which in collaboration with the FBS modeler incorporates the QFD method into conceptual design (Yoshioka *et al.* 1995).
- A symbolic math system, Mathematica, to be used for building various kinds of engineering models. This resulted in useful applications, because many of simple engineering models, such as a Bond Graph modeler to evaluate dynamic behavior of design objects (Ishii *et al.* 1995) and a beam model to evaluate strength and deformation (Yoshioka *et al.* 1993).
- QPAS (Qualitative Process Abduction System) for supporting synthesis that can generate behaviors and states from functional information (Ishii *et al.* 1994). This system can be used for functional design in which behaviors and states of a design solution must be reasoned out from functional specifications.
- A system for generating sequence control software for mechatronics machines (Shimomura *et al.* 1993; Tomiyama *et al.* 1993). This system takes functional information about a mechatronics machine and generates qualitative behavioral information using QPAS. Then, this qualitative control sequence is augmented with quantitative data, such as sensor layout and part dimensions, obtained through the metamodel mechanism from a geometric modeler, etc.
- Systems for total maintenance. First, there is a model based fault diagnostic system. The metamodel system can convert an object model, described according to QPT in the metamodel, to a device oriented one for the confluence type qualitative reasoning system. This device oriented model can be used for repair planning, since it can represent faulty states and a normal state. This resulted in the development of a self-maintenance machine. Also, tools for reliability design, including Fault Tree Analysis and evaluation of reliability are incorporated.
- A Petri net simulator for discrete event simulation. This can be used for simulating, for instance, manufacturing processes and control logic of mechatronics systems.

6 CONCLUSIONS

This paper proposed and discussed the concept of knowledge intensive engineering that is a new way of engineering activities in various product life cycle stages conducted with more knowledge in a flexible manner to create more added value. It also described a computational framework for knowledge intensive engineering called KIEF we are currently developing.

Knowledge intensive engineering requests that various kinds of engineering knowledge must be systematically organized and installed in a computational framework such as KIEF.

The architecture of KIEF should be the cooperative multiple intelligent agent architecture. A critical issue here is knowledge systematization that includes setting up a view, articulation, codification, verification, crystallization, and reusing and sharing. This can be achieved by having not only a common knowledge description format but also terminological, taxonomical, and ontological level standardization. The net result includes not only sharable knowledge but also a knowledge standard. In Chapter 5, we discussed the kernel knowledge representation and modeling methodologies of KIEF. Its architecture allows to plug in external intelligent agents.

Knowledge intensive design on the framework does not only mean advanced automation of design but also allows designers to arrive at creative, innovative design. The design of self-maintenance machines demonstrates that knowledge intensive engineering can lead to more creative, innovative design with increased added value, which is a knowledge intensive machine and knowledge intensive maintenance.

Although practical use of KIEF is yet to be carried out, the system received promising responses from designers and engineers who participated experimental trials (Ranta *et al.* 1995; Umeda *et al.* 1994a). One of the advantages of designing on KIEF is that it is a truly integrated computer-aided engineering framework that allows smart model management. However, to do so, we need to prepare all the concepts (i.e., entities, relations, physical phenomena, etc.) needed in the models and these concepts are clearly understood by the designers who have to maintain correspondences among concepts. Unfortunately this preparatory knowledge acquisition process is laborious. Models can be almost automatically generated and maintained, only after such concepts are ready in the knowledge base. This implies that commonsense reasoning based on VLKB should be considered to reduce the effort. Perhaps, the most comparable work to KIEF is the PACT project. Their ontology is based on Ontolingua and the knowledge is represented in KIF. However, they put more emphasis on providing a computational collaborative concurrent engineering environment and do not solve the knowledge acquisition problem, either.

Another advantage of KIEF is that it integrates functional knowledge in a comprehensive manner, which is truly unique among other systems developed more or less in the same philosophy. For instance, de Vries developed the MAX system aiming at a mechatronic model building environment (de Vries 1994). The primary ontology of this system is the Bond Graph methodology, which is used for the Schemebuilder system developed by Oh *et al.* (1994). The FBS modeling is more general than these systems based on Bond Graphs.

Future work includes further plugging-in external modelers, expanding the knowledge base, and applications and evaluation of KIEF in a practical context.

7 ACKNOWLEDGMENT

This paper represents and summarizes a long research effort of our group at the University of Tokyo. We are grateful to Professor Hiroyuki Yoshikawa (currently President of the University of Tokyo) who outlined and initiated the research two decades ago. We would like to thank members of our group who greatly contributed to the implementation of the system and helped us to prepare this paper.

8 REFERENCES

Cutkosky, M., Engelmore, R., Fikes, R., Genesereth, M., Gruber, T., Mark, W., Tenenbaum, J., and Weber., J. (1993) PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, **26** (1), 28-37.

de Vries, T.J.A. (1994) *Conceptual design of controlled electro-mechanical systems.* Ph.D. Thesis, University of ., Enschede, the Netherlands.

Forbus, K. D. (1984) Qualitative process theory. *Artificial Intelligence*, **24** (3), 85-168.

Genesereth, M. R. and Fikes, R.. (1990) *Knowledge interchange format, version 2.2 reference manual.* Computer Science Department, Stanford University, Stanford, CA, USA. Technical Report Logic-90-4.

Gruber, T. R. (1993) *Towards principles for the design of ontologies used for knowledge sharing.* Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA. Technical Report KSL 93-04.

Ishii, M., Sekiya, T., and Tomiyama, T. (1995) A very large-scale knowledge base for the knowledge intensive engineering framework. In *Towards Very Large Knowledge Bases* (ed. N.J.I. Mars), IOS Press, Amsterdam, Oxford, Tokyo, Washington, D.C., 123-131.

Ishii, M., Tomiyama, T. and Yoshikawa, H. (1994) Synthetic reasoning method for conceptual design. In *Towards World Class Manufacturing* (eds. M.J. Wozny and G. Olling), IFIP Transactions B-17, North-Holland, Amsterdam, 3-16.

ISO. (1990) *STEP (Standard for The Exchange of Product model data), Draft Proposal.* ISO/TC184/SC4.

Kiriyama, T., Tomiyama, T., and Yoshikawa, H.. (1991) The use of qualitative physics for integrated design object modeling. In *Design Theory and Methodology -DTM '91-* (ed. L. Stauffer), ASME, New York, 53-60.

Kiriyama, T., Tomiyama, T., and Yoshikawa, H. (1992) Qualitative reasoning in conceptual design with physics features. In *Recent Advances in Qualitative Physics*, (eds. B. Faltings and P. Struss), The MIT Press, Cambridge, MA, USA, 375-386.

Lenat, D. B. (1995) Steps to sharing knowledge. In *Towards Very Large Knowledge Bases*, (ed. N. J. I. Mars), IOS Press, Amsterdam, Oxford, Tokyo, Washington, D.C., 3-6.

Lenat, D. B. and Guha, R. V. (1989) *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* Addison-Wesley, Reading, MA, USA.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. R. (1991) Enabling technology for knowledge sharing. *AI Magazine*, **12** (3), 36-56.

Oh, V., Langdon, P., and Sharpe, J. E. E. (1994) Schemebuilder: An integrated environment for product design. In *Lancaster International Workshop on Engineering Design CACD '94, in Lancaster University, Lancaster, UK*, (eds. V. Oh and J. E. E. Sharpe).

Ohtsubo, H., Kawamura, Y., and Kubota, A. (1993) Development of the object-oriented finite element modeling system - MODIFY. *Engineering with Computers*, **9** (4), 187-197.

Ranta, M., Mantyla, M., Umeda, Y., and Tomiyama, T. (1995). Integration of functional and feature-based product modeling - The IMS/GNOSIS experience. *Computer-Aided Design* (in printing).

Shimomura, Y., Sakao, T., Ohmichi, K., Widmer, T., Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1993) Model-based automatic generation of control sequence from design information. In *Proceedings of the Eighth Annual Meeting of the American Society of for Precision Engineering in Seattle, WA, USA*, ASPE, Raleigh, NC, USA, 495-498.

Tomiyama, T. (1994) From general design theory to knowledge-intensive engineering. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, **8** (4), 319-333.

Tomiyama, T. (1995) A Japanese view on concurrent engineering. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM), Special Issue on Concurrent Engineering*, **9** (2), 69-71.

Tomiyama, T. and Baba, Y. (1993) Artifactual engineering and the post-mass production paradigm. In *the First International Symposium on Research into Artifacts, in Tokyo, RACE*, the University of Tokyo, Tokyo, Japan, 16-23.

Tomiyama, T., Kiriyama, T., Takeda, H., Xue, D., and Yoshikawa, H. (1990) Metamodel: A key to intelligent CAD systems. *Research in Engineering Design*, **1** (1), 19-34.

Tomiyama, T., Kiriyama, T., and Umeda, Y. (1994) Toward knowledge intensive engineering. In *Knowledge Building and Knowledge Sharing*, (eds. K. Fuchi and T. Yokoi), IOS Press, Amsterdam, Oxford, Washington, D.C., 308-316.

Tomiyama, T., Umeda, Y., and Yoshikawa, H. (1993) A CAD for functional design. *Annals of CIRP*, **42** (1), 143-146.

Tomiyama, T., Xue, D., Umeda, Y., Takeda, H., Kiriyama, T., and Yoshikawa, H. (1992) Systematizing design knowledge for intelligent CAD systems. In *Human Aspects in Computer Integrated Manufacturing*, (eds. G. Olling and F. Kimura), IFIP Transactions B-3, North-Holland, Amsterdam, 237-248.

Umeda, Y., Ishii, M., Yoshioka, M., and Tomiyama, T. (1994a) Experimental use and extension of the FBS modeler. In *Representing and Reasoning About Device Function, AAAI-94 Workshop*, AAAI, 154-160.

Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1990) Function, behavior, and structure. In *Applications of Artificial Intelligence in Engineering V, Proceedings of the Fifth International Conference, 1: Design*, (ed. J. S. Gero), Springer-Verlag, Berlin, 177-193.

Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1991) A design methodology for a self-maintenance machine. In *Design Theory and Methodology –DTM '91 –*, (ed. L. A. Stauffer), ASME, 143-150.

Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1992) A design methodology for a self-maintenance machine based on functional redundancy." In *Design Theory and Methodology –DTM '92 –*, (eds. D. Taylor and L. Stauffer), ASME, 317-324.

Umeda, Y., Tomiyama, T., Yoshikawa, H., and Shimomura, Y. (1994b) Using functional maintenance to improve fault tolerance. *IEEE Expert, Intelligent Systems & Their Applications*, **2** (3), 25-31.

Veth, B. (1987) An integrated data description language for coding design knowledge. In *Intelligent CAD Systems I – Theoretical and Methodological Aspects*, (eds. P. J. W. ten Hagen and T. Tomiyama), Springer-Verlag, Berlin, 295-313.

Xue, D., Takeda, H., Kiriyama, T., Tomiyama, T., and Yoshikawa, H. (1992) An intelligent integrated interactive CAD – A preliminary report. In *Intelligent Computer Aided Design*, (eds. D. Brown, M. Waldron, and H. Yoshikawa), IFIP Transactions B-4, North-Holland, Amsterdam, 163-192.

Yoshikawa, H., Tomiyama, T., Kiriyama, T. and Umeda, Y. (1994) An integrated modeling environment using the metamodel. *Annals of the CIRP*, **43** (1), 121-124.

Yoshioka, M., Nakamura, M., Tomiyama, T., and Yoshikawa, H. (1993) Design process model with multiple design object models. In *Design Theory and Methodology -DTM '93,* (eds. T. K. Hight and L. A. Stauffer), ASME, 7-14.

Yoshioka, M., Oosaki, M., and Tomiyama, T. (1995) An application of quality function deployment to functional modeling in a knowledge intensive design environment. Acepted for publication in this volume.

9 BIOGRAPHY

Dr. Tetsuo Tomiyama has been Associate Professor at the Department of Precision Machinery Engineering, the University of Tokyo, since 1987. From 1985 to 1987, he worked at the Centre for Mathematics and Computer Science in Amsterdam. He received his doctor's degree in precision machinery engineering from the Graduate School of the University of Tokyo in 1985. His research interest includes design theory and methodology, knowledge intensive engineering, applications of qualitative physics, large scale engineering knowledge bases, and soft machines (self-maintenance machines and cellular machines). He is a member of the IFIP Working Group 5.2.

Dr. Yasushi Umeda has been a lecturer of Inverse Manufacturing Laboratory of the Faculty of Engineering, the University of Tokyo, since 1995. From 1992 and 1995, he was research associate at the Department of Precision Machinery Engineering of the University of Tokyo. He received his doctor's degree in precision machinery engineering from the Graduate School of the University of Tokyo in 1992. His research interests include soft machines (e.g., self-maintenance machines and cellular machines), green life cycle design, intelligent CAD for mechanical design, and functional reasoning.

Masaki Ishii is a Ph.D. Candidate at the Department of Precision Machinery Engineering Department of the University of Tokyo. He received his B.Sc. and M.Sc. degrees in precision machinery engineering also from the University of Tokyo in 1991 and 1993. His current research interest includes knowledge sharing, model-based reasoning, knowledge intensive engineering, and large-scale engineering knowledge bases.

Masaharu Yoshioka is a Ph.D. student in the Department of Precision Machinery Engineering, The University of Tokyo. His research interest includes design process modeling, functional modeling, and knowledge intensive engineering. He will obtain his Ph.D. in precision machinery engineering in March 1996.

Dr. Takashi Kiriyama graduated from Department of Precision Machinery Engineering, Faculty of Engineering, the University of Tokyo in 1986. He obtained his Doctor's degree in Engineering from the Graduate School of the University of Tokyo in 1991. In the same year he started to work at the Department as Research Associate. In 1992 he became Lecturer at Research into Artifacts, Center for Engineering (RACE), the University of Tokyo, and has been Associate Professor since 1995. His research interest includes artifactual engineering, network-based collaborative design, qualitative physics, and design of micromechanisms.